2018

Switching Neural Network Systems for Nonlinear Tracking

Manoj Ghimire
Wright State University

SWITCHING NEURAL NETWORK SYSTEMS FOR NONLINEAR TRACKING

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Electrical Engineering

by

MANOJ GHIMIRE
B.E., Tribhuvan University, 2012

2018
Wright State University

Wright State University
GRADUATE SCHOOL

December 11, 2018

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY Manoj Ghimire ENTITLED Switching Neural Network Systems for Nonlinear Tracking BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Master of Science in Electrical Engineering.

Joshua N. Ash, Ph.D.
Thesis Director

Fred D. Garber, Ph.D.
Interim Chair, Department of Electrical Engineering

Committee on
Final Examination

Joshua N. Ash, Ph.D.

Arnab K. Shaw, Ph.D.

Fred D. Garber, Ph.D.

Barry Milligan, Ph.D.
Interim Dean of the Graduate School

ABSTRACT

Ghimire, Manoj. M.S.E.E., Department of Electrical Engineering, Wright State University, 2018. *Switching Neural Network Systems for Nonlinear Tracking.*

In this thesis, we consider the problem of tracking in complex nonlinear dynamical systems. While the Kalman filter is known to be the mean-squared error optimal tracker under linear dynamics and linear measurements, more sophisticated models and algorithms are required for complex dynamics. Here, we consider switching systems where the dynamical properties vary ("switch modes") over time. For example, the dynamics of a vehicle may switch as it transitions from interstate to urban conditions, human speech dynamics switch as speakers change, and stock market dynamics switch with discrete news events. In this work, we use mode-dependent neural networks to capture different nonlinear dynamics in a given system, and we developed a new algorithm, dubbed the Switching Neural Network Tracker (SNNT), to track modes and states over time. The proposed Bayesian system includes Markovian dynamics to model mode transitions and employes the Unscented Transform to mitigate computational complexity while estimating posterior probabilities. Examples using synthetic robot motion data and measured honeybee dance data demonstrate accurate mode identification and a dramatic reduction in state estimation error relative to non-switching systems.

Contents

List of Figures

Acknowledgments

I am indebted to the support of my thesis adviser Dr. Josh Ash throughout my journey in the Graduate school, from the beginning to the last semester, thank you. I would also like to thank my thesis committee members, Dr. Arnab K. Shaw and Dr. Fred D. Garber for taking time in reviewing my thesis.

I would also like to thank all my friends for their support for making my experience at graduate school such a wonderful one.

Last but not least, I would like to thank my parents, my sister and my brother-in-law for their continuous support throughout this journey.

Introduction

1.1 Switching Dynamical Systems

In this thesis, we consider the problem of tracking in complex nonlinear dynamical systems. State-space tracking typically involves a dynamical model

$$x_{k+1} = F(x_k) \tag{1.1}$$

that attempts to capture physical processes that evolve a state x_k at time k to state x_{k+1} at the next time instant. For example, x_k may represent the 2D position of a vehicle and $F()$ encodes temporal motion dynamics, or x_k could represent a scalar temperature while $F()$ captures atmospheric effects leading to temperature changes over time. Tracking algorithms exploit the dynamical model in order to predict future states from previous states, or measurements derived from previous states. Dynamical systems are categorized based on their input-output relationship $F()$ and follow either a linear model or non-linear model. Linear Dynamical Systems (LDS) are systems in which the system parameters follow linear dynamical properties, i.e. the innovation from one state to another state and the measurement model are linear expressions of the current state. A nonlinear dynamical system has non-linearity in its state update and measurement models.

When dynamics are linear with Gaussian perturbations

$$x_{k+1} = F(x_k) = Ax_k + n_k \tag{1.2}$$

and available measurements are linear-Gaussian $y_k = Hx_k + w_k$, the Kalman filter is known to be the mean-square error optimal tracking algorithm [1]. In many cases, however, linear models do not suffice, and for complicated dynamics multiple dynamical models are more appropriate. For example, a physical system may be better modeled using two mathematical models

$$x_{k+1} = F_1(x_k) \qquad \text{under condition/mode 1} \tag{1.3}$$

$$x_{k+1} = F_2(x_k) \qquad \text{under condition/mode 2,} \tag{1.4}$$

where each model is more appropriate under distinct operational conditions. For example in tracking a vehicle, one model may be more appropriate for interstate driving while another better suited for urban conditions. We consider a hidden mode variable z_k that indicates the most appropriate model at time k: $x_{k+1} = F_{z_k}(x_k)$. The progression of $z_0, z_1, ...$ through time indicates when the system "switches modes." If the mode z_k is known, then tracking can be improved by using a dynamical model tailored for the specific conditions at the time. Generally, the modes $\{z_k\}$ are unknown and need to be estimated, however beyond improved state ($\{x_k\}$) tracking, knowledge of the modes is useful because they indicate broad conditions of the system, e.g. interstate vs. rural conditions in automotive tracking, speaker 1 vs. speaker 2 in speech denoising, etc. These are frequently of interest in their own right. Additionally, a system that can switch between different simpler dynamical systems can effectively capture the dynamics of complex nonlinear dynamical systems [16] and these types of switching systems are highly valuable as they can model many real systems and have applications in a vast array of fields from the stock market [11] to meteorology [17].

1.1.1 Examples and Applications of Switching Dynamical Systems

Here we review a few examples where switching behavior can be applied and exploited in practical applications.

- **Stock Market Prediction:** The stock market is considered a highly volatile system with many factors changing its state every second. Market sentiments, news, the discussion in social media, political and social decision all affect the stock market trends. Predicting the stock market is a challenging subject due to the presence of measurable and random components in varying proportions [24], [27]. Switching models for the stock market could indicate bull or bear markets, or denote sentiment switches corresponding to discrete news events. An efficient switching model to optimize the stock selection scheme and return rate would be of great use for stock investors and players in the financial sectors as well. Hidden Markov models have been applied for selecting stock with high resulting return in the portfolio [22], [21].

- **Robotics:** A goal of current research in robotics is to develop autonomous robotic systems and be able to reproduce characteristic features of living creatures like advanced mobility, a sense of location etc. To achieve complicated navigational and movement task that living beings can easily do, robotic systems should be able to identify its bearing or mode at any instance of time. There are various algorithms that have been used to track modes of the robotic systems [2], [4], [3]. Figure 1.1 represents a trajectory taken by a robot along with different modes at every time instance k. In this example, the robot can switch between modes $z_k \in \{0, 1\}$ at time k. Mode $z_k = 0$ represents a robotic system going through a circular path and $z_k = 1$ represents it going through the linear path. This system can be used along with a probabilistic model to predict the next mode that the system may acquire with the passage of time. This setup can effectively be used for path planning, obstacle avoidance, and various other autonomous tasks.

• **Speech Recognition:** Speech processing and recognition are actively researched field [8], [19]. In real-life, speech data from voice recording devices are corrupted by noise and extraction and modeling of a true speech signal from the noise-corrupted signal is valuable in many fields like speech translation, voice-enabled devices. Switching algorithm with different statistics for each distinct modes have been applied to a noisy non-stationary speech signal to accurately model the clean underlying speech signal and noise with superior performance [19].

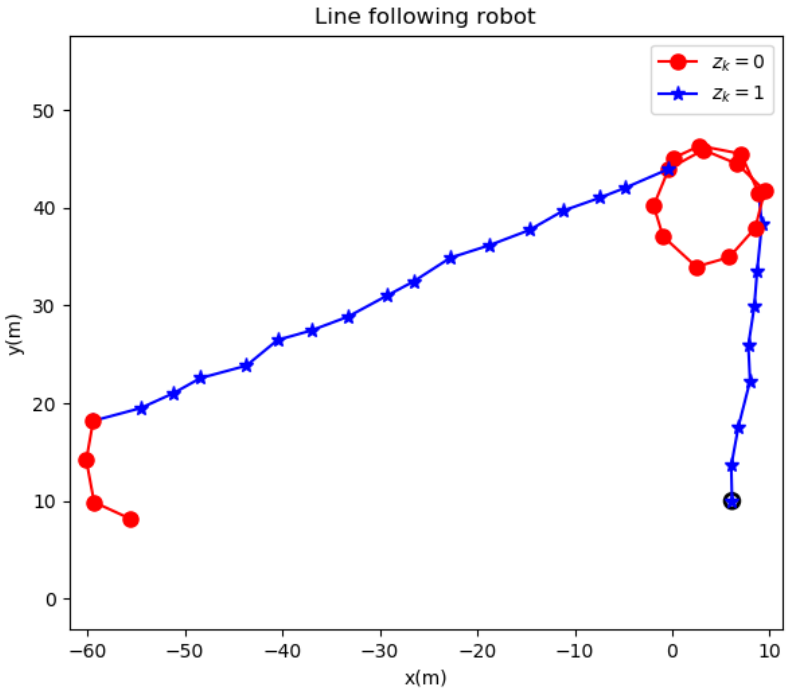
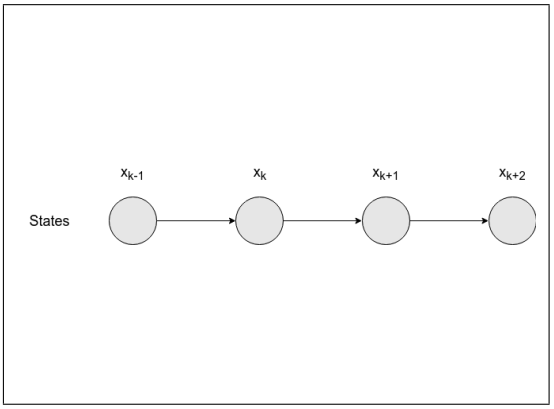
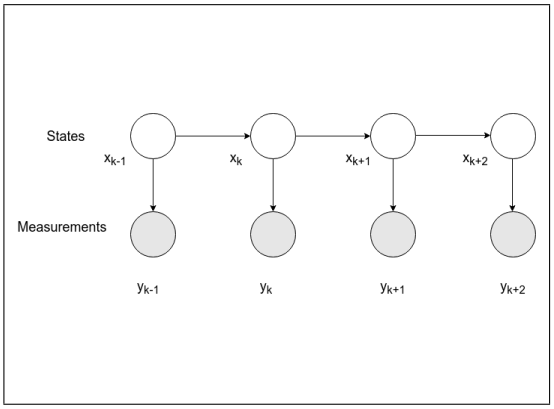


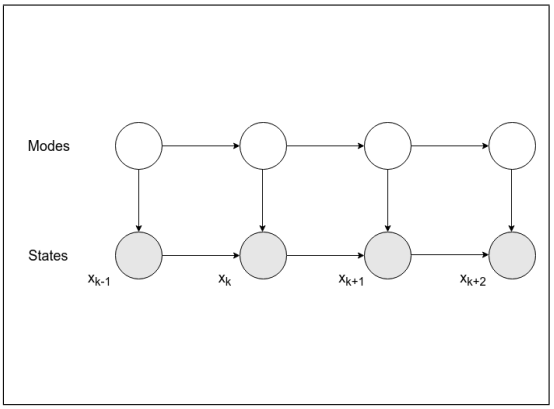
Figure 1.1: Example of a wheeled robot's trajectory with different modes of motion. Mode $z_k = 0$ represents a turning motion pattern, and mode $z_k = 1$ represents straight-line motion.



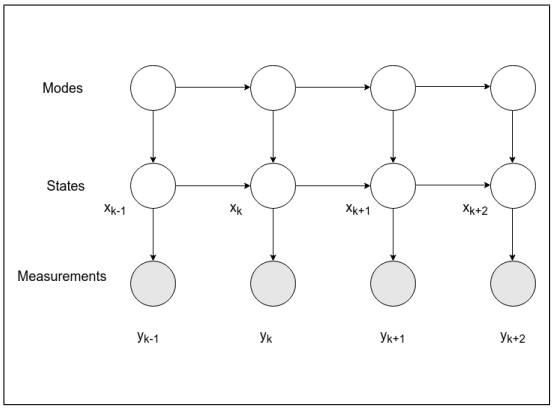
(a) Autoregressive model



(b) Dynamical system model



(c) Switching autoregressive model



(d) Switching dynamical system model

Figure 1.2: Taxonomy of dynamical models

- **Basketball Gameplay Modeling:** The movement path individual basketball players take in a team can also be considered as a complex dynamical system. Each player gets a ball and plays in their area and passes it to a different player in the different area. Switching modes correspond to defensive vs. offensive play and also relate to different physical locations on the court [16].

1.2 Contributions and Organization of the Thesis

In Figure 1.2 we present a taxonomy of dynamical models (which are described in detail in Section 2.1). The Kalman filter and other single mode tracking systems are represented by the classic dynamical system depicted in Figure 1.2(b). In this work, we consider switching dynamical systems as depicted in Figure 1.2(d). In the switching model, an additional layer of hidden variables, referred to as modes, is present and dictates how the state variables evolve. Switching dynamical systems have been extensively studied [9, 7, 23, 20], however the primary focus has been on switching linear systems, where dynamics are simply specified by different transition matricies. Here, we focus on non-linear switching systems where each individual mode is itself non-linear.

This thesis contributes to the tracking literature in the following ways.

1. We developed a novel switching dynamical system model employing distinct neural networks to capture mode-dependent non-linear state dynamics. A Markov chain controls the switching dynamics between the modes.

2. We developed a new algorithm to provide posterior estimates of both the modes and states in the above model. The algorithm is referred to as the Switching Neural Network Tracker and utilizes the Unscented Transform to perform Bayesian inference with minimal computational complexity.

3. We applied our algorithm to synthetic robot motion data and measured honeybee dance data. These examples demonstrated accurate mode identification and dramatic reduction in state estimation error relative to non-switching systems.

The remainder of this thesis is organized as follows. In Chapter 2 background technical details that lay the foundation for the main contribution of the thesis are presented. Specifically, we review dynamical models and traditional tracking algorithms such as the

Kalman filter and Unscented Kalman filter for linear and non-linear tracking, respectively. In Chapter 3 we first present the new switching dynamical system model employing neural networks and Markovian mode dynamics, and then we develop the new tracking algorithm dubbed the Switching Neural Network Tracker (SNNT). Results are presented in Chapter 4 for both synthetic and measured data. In this chapter we also present a machine learning approach that uses training data to estimate the optimal transition matrix for a Kalman filter. This optimal Kalman filter then serves as a baseline of comparision for our SNNT algorithm. Finally, conclusions and avenues of future work are discussed in Chapter 5.

Background

In this chapter we review background material necessary to develop our SNNT model and establish its place in the context of existing literature. We begin by reviewing a taxonomy of dynamical models in Section 2.1 and proceed by considering tracking algorithms in Section 2.2.

2.1 Dynamical Models

A complex dynamical system can be modeled by a simpler system that transition from one dynamical model to another from a set of dynamical models [7]. Common models used for discrete systems are the subset of probabilistic models like Hidden Markov Model or State Space model. Hidden Markov model is a model in which hidden states are represented by random variables and posterior on current state probability distribution is obtained using the previous state and probabilistic transition matrix. State space model is a model in which hidden states are represented by real-valued variables and the current state of the system is obtained by applying a system equation and noise model to the previous state of the system.

Real world systems have processes with nonlinear and dynamical parts so they cannot be modeled completely by pure probabilistic or linear system. These systems are a mixture of different simpler models. An example of a real-world system consisting of different models is Honeybee dance. Honeybee moves with the different pattern to communicate the locations of resources to the colony. They use waggle movement to signify location farther

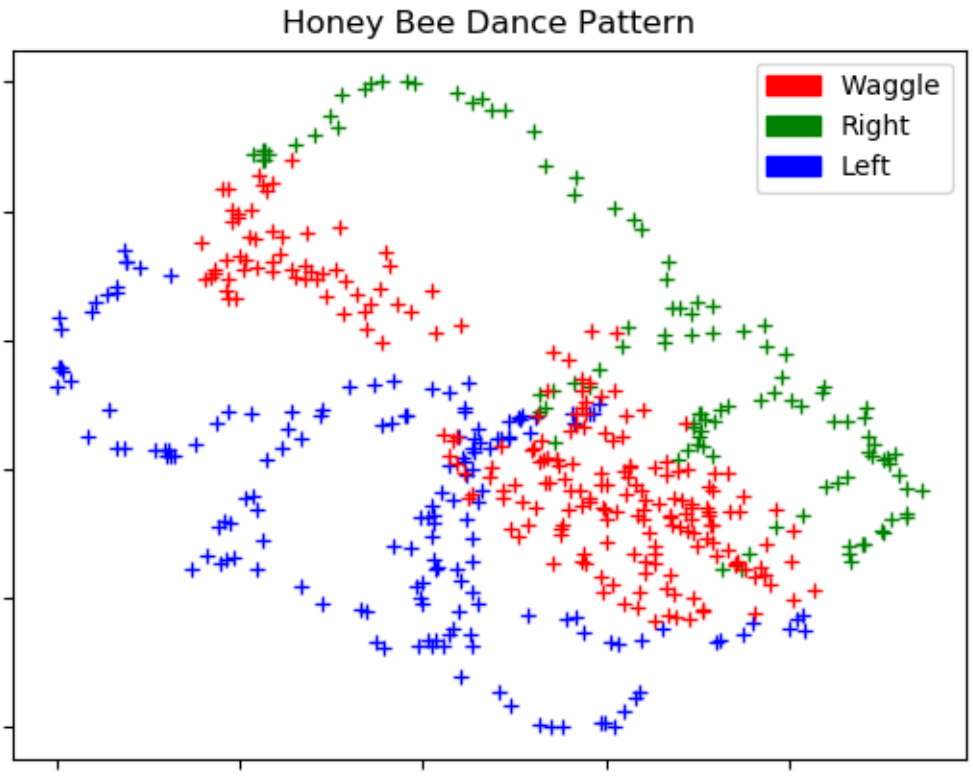


Figure 2.1: Honeybee dancing pattern

away from the colony and circular motion for resources near the colony.

In Figure 2.1 there are three movement patterns: waggle, right and left. Each of the movement patterns can be considered as a nonlinear dynamical system. This way the motion patterns can be modeled by the mixture model and switching can be done between different models.

Various models currently exist in literature to represent the dynamical systems. We will discuss four of the most important models below.

2.1.1 Autoregressive Model

An autoregressive model (AR model) is a mathematical model which is a linear combination of past observations and noise. For a time series x_k at time k with past state x_{k-1}. AR model is mathematically expressed as

$$x_k = \sum_{i=1}^{j} a_i x_{k-i} + v_k \tag{2.1}$$

where a_i and j are the autoregressive coefficient and autoregressive order, respectively. v_k is assumed to be white noise following a normal distribution with mean 0 and variance σ^2 satisfying the condition of independence that E[$v_k v_j$] = 0, for $k \neq j$ and E[$v_k x_j$] = 0, for $k > j$.

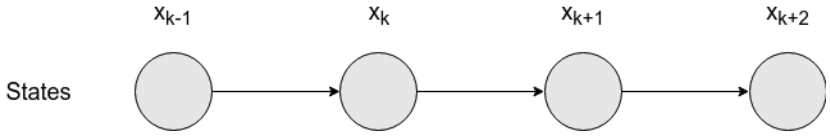


Figure 2.2: Autoregressive Model

2.1.2 Dynamical System Model

A dynamical system model represent Kalman Filter. For a series x_k at time k with past state x_{k-1} dynamical system model is expressed as

$$x_k = Ax_{k-1} + Bv_{k-1} + w_k, \tag{2.2}$$

where x_k is the state vector, A is the state transition matrix, x_{k-1} is the state vector at time $k-1$, v_{k-1} is the system noise vector, w_k is the system innovation noise vector. The

measurement model is expressed as

$$y_k = Hx_k + u_k, \tag{2.3}$$

where y_k is the measurement vector, H is the observation matrix, x_k is the state vector and u_{k-1} is the measurement noise vector.

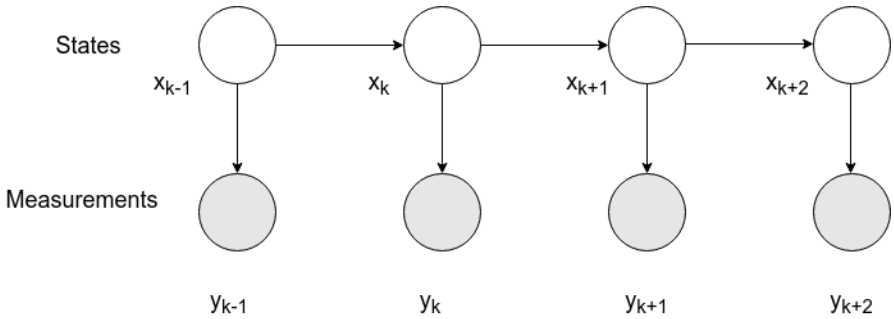


Figure 2.3: Dynamical System Model

2.1.3 Switching Autoregressive Model

A switching autoregressive model (Figure 2.4) is an Autoregressive Model with discrete mode variable z_k for each time instance k. The mode z_k can be one of the modes from a set of S different modes each with different AR parameter settings. The set of modes are considered to be Markovian with transition probability $p(z_k|z_{k-1})$. The probability p for next switching mode z_k being j is given by

$$p(z_k = j|z_{k-1}, x_{k-1}, x_k) \propto p(z_k = j|z_{k-1})p(x_k|x_{k-1}, z_k) \tag{2.4}$$

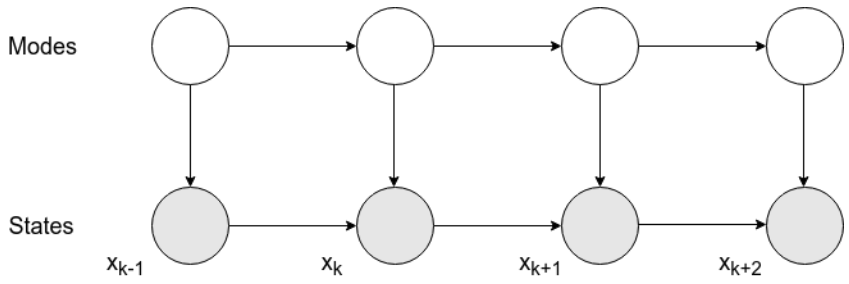


Figure 2.4: Switching AR Model

2.1.4 Switching Dynamical System Model

The switching dynamical system model is similar to the switching AR model, however the states are now hidden and a state-dependent observation is measured. At each time instant we have $z_k \in S$ to indicate the mode at time k. As before, the modes follow the Markov chain. The measurement at each time sequence is measurement is represented by y_k. The probability of next mode being z_k given the observation y_k is proportional to the product probability of z_k and likelihood function of y_k given z_k. Mathematically it is expressed as

$$p(z_k|y_k) \propto p(z_k)\, \boldsymbol{L}(y_k|z_k) \tag{2.5}$$

For mean μ_{y_k} and covariance $\boldsymbol{C}_{y_k}$ the marginalized likelihood function is approximated by the Gaussian approximation

$$\boldsymbol{L}(y_k|z_k) \approx \mathcal{N}(y_k; \mu_{y_k}, \boldsymbol{C}_{y_k}) \tag{2.6}$$

2.2 Traditional Tracking Filters

In this age of technology and advancement, information is the most powerful tool at hand. In real-life situations most often information is corrupted either by noise or the desired information is mixed with undesired features or both. So it is of great importance to be

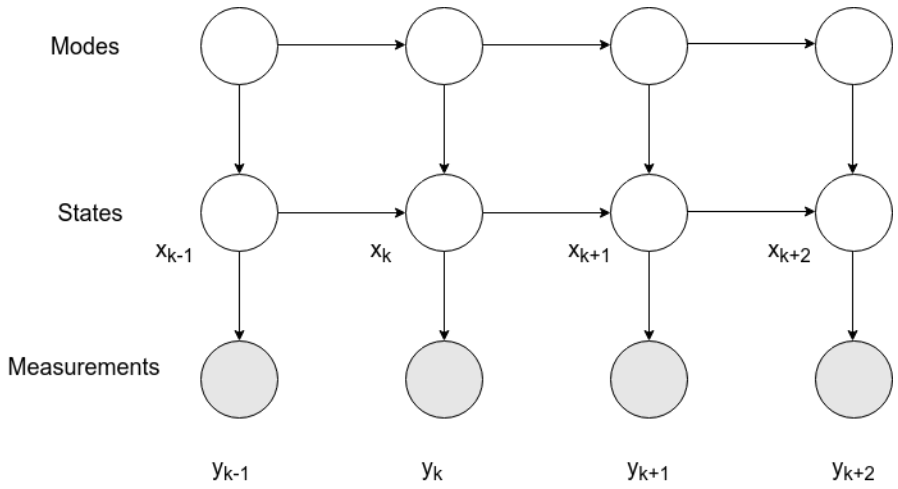


Figure 2.5: Switching Dynamical System Model

able to extract useful information from the corrupted information. The technical term for this process is called filtering. Filtering is a way to obtain useful information hidden in a signal corrupted by random variables or undesired variables. In signal processing and engineering in general, filtering is just processing the signal performing frequency domain analysis or using advanced filtering algorithms to estimate the signal buried in the noisy message and minimize the effect of noise in the signal.

Use cases of filtering are everywhere, they are used from the financial sector, telecommunication to space exploration [10]. In finance, advanced filtering algorithms have been used to track the movement of the market and predict the bull or bearish market trend in stock. In telecommunications, filtering is used to minimize the effect of noise that the signal acquires during transmission from transmitting device to a transmission medium. A filtering algorithm was used to track the trajectory of the spaceship during Apollo project [10] and it was considered a really great achievement to humankind.

2.2.1 Kalman Filtering

Kalman filter [14] is one of the most widely used linear optimal estimation algorithm. Basically, it consists an array of mathematical equations that predicts and corrects the estimates based on the measurement data. This algorithm is considered most optimal for a linear

system as it has the minimum estimated error covariance among various other estimation algorithms with the same noise models. To estimate the hidden states of the signal mixed with the noise it uses knowledge of the system and measurement device, statistical description of the system, system noise, measurement noise and initial condition of the system [18]. At each time instance, the Kalman filter predicts the new state of the system using the previous state and noise model then measurement data are fused to the predicted state to correct the prediction and to produce a more accurate estimate of the state.

Kalman filter state equation is represented as:

$$x_k = Ax_{k-1} + Bu_{k-1} + w_k \tag{2.7}$$

where x_k is the state of the system at time k, x_{k-1} is the state of the system at time $k-1$, A is the state transition matrix which relates the state of the system at time $k-1$ to the state at time k without system noises, u_{k-1} is the system noise, w_k is the process noise of the system at time k.

The measurement model in Kalman filter is linear and it can be represented as

$$y_k = Hx_k + v_k \tag{2.8}$$

Where y_k is the measurement quantity, H is the transformation matrix that transforms state to measurement, x_k is the state variable and v_k is the measurement noise.

The process noise and measurement noise are assumed to be independent of each other and have a normal distribution.

$$w \sim \mathcal{N}(0, Q) \tag{2.9}$$

$$v \sim \mathcal{N}(0, R) \tag{2.10}$$

where Q and R are the innovation and measurement noise covariance matrix respectively.

Mathematically, $a\ priori$ prediction error e_k^- and $a\ posteriori$ prediction error e_k are computed as follows:

$$\begin{aligned} e_k^- &= x_k - \hat{x}_k^- \\ e_k &= x_k - \hat{x}_k \end{aligned} \tag{2.11}$$

where $\hat{x}_k^-$ is the $a\ priori$ of the state estimate at time step k and $\hat{x}_k$ is the $a\ posteriori$ of the state estimate with the knowledge of measurement y_k at time step k.

Using Equation 2.11, $a\ priori$ error covariance P_k^- and $a\ posteriori$ error covariance P_k are computed as

$$\begin{aligned} P_k^- &= E[e_k^- {e_k^-}^T] \\ P_k &= E[e_k {e_k}^T] \end{aligned} \tag{2.12}$$

Kalman filter prediction equations are defined as follows:

$$\begin{aligned} \hat{x}_k^- &= A\hat{x}_{k-1} + Bu_{k-1} \\ P_k^- &= AP_{k-1}A^T + Q \end{aligned} \tag{2.13}$$

Update equations are defined as follow:

$$\begin{aligned} K_k &= P_k^- H^T (HP_k^- H^T + R)^{-1} \\ \hat{x}_k &= \hat{x}_k^- + K_k(y_k - H\hat{x}_k^-) \\ P_k &= (I - K_k H)P_k^- \end{aligned} \tag{2.14}$$

Where $\hat{x}_k, P_k, K_k, I$ are the state estimation, estimated error covariance matrix, Kalman gain matrix respectively at time instance k and I is an identity matrix. Kalman filter is a recursive estimation process where the estimate $\hat{x}_k$ from the previous iteration is again

passed into the Equation 2.13 and 2.14 giving a new estimate for the current iteration.

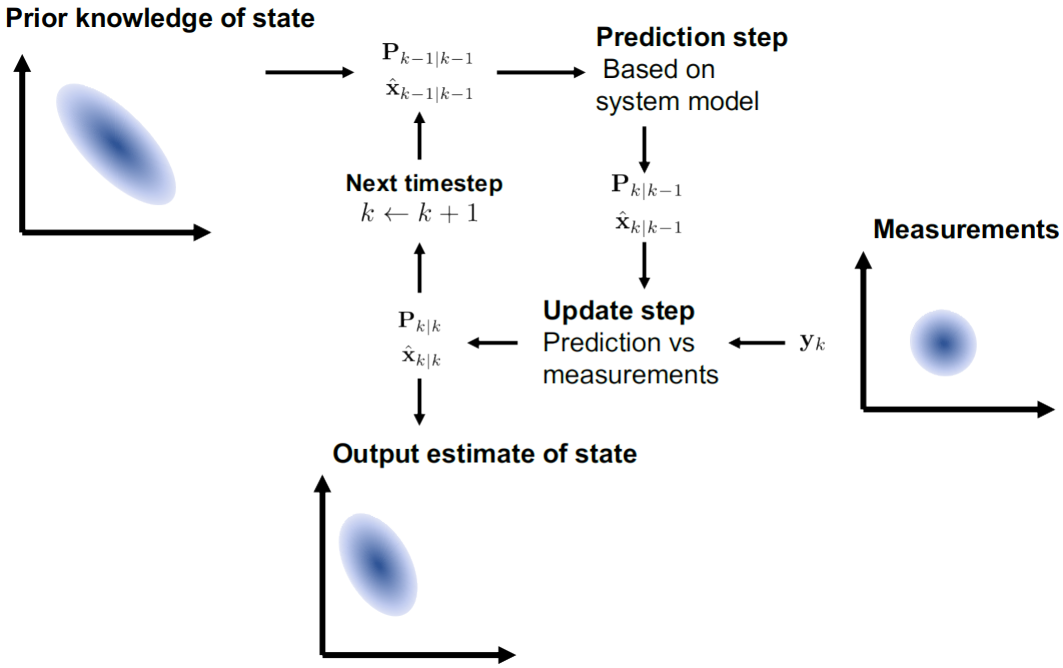


Figure 2.6: Kalman Filter Algorithm

2.2.2 Unscented Kalman Filtering

Kalman filter is regarded as a Minimum Mean Squared Error (MMSE) estimator for linear systems. But for the nonlinear case, many different filtering algorithms have been developed one such algorithm is Unscented Kalman Filter [13, 25, 26]. Unscented Kalman Filter (UKF) approximates the states of a system using carefully selected weighted points, called sigma points, which has the same distribution as the original state. The sigma points are assumed to follow a Gaussian distribution. UKF uses the unscented transform to calculate the first and second order moments of the system undergoing nonlinear transformation using these sigma points and measurement is used to improve the accuracy of the estimate. This process is repeated at every time step to get the estimates of the signal at different time instance.

Unscented Transform

Unscented Transform is an algorithm for calculating the first and second moments of a random variable which undergoes a transformation in nonlinear fashion. It chooses weighted points, called the sigma points, having the same mean and covariance as the original distribution and applies nonlinearities to these sigma points. Figure 2.7 shows the working of the unscented transform. The first figure shows the true sampling and transformation process and the second figure shows the method used by the unscented transform process.

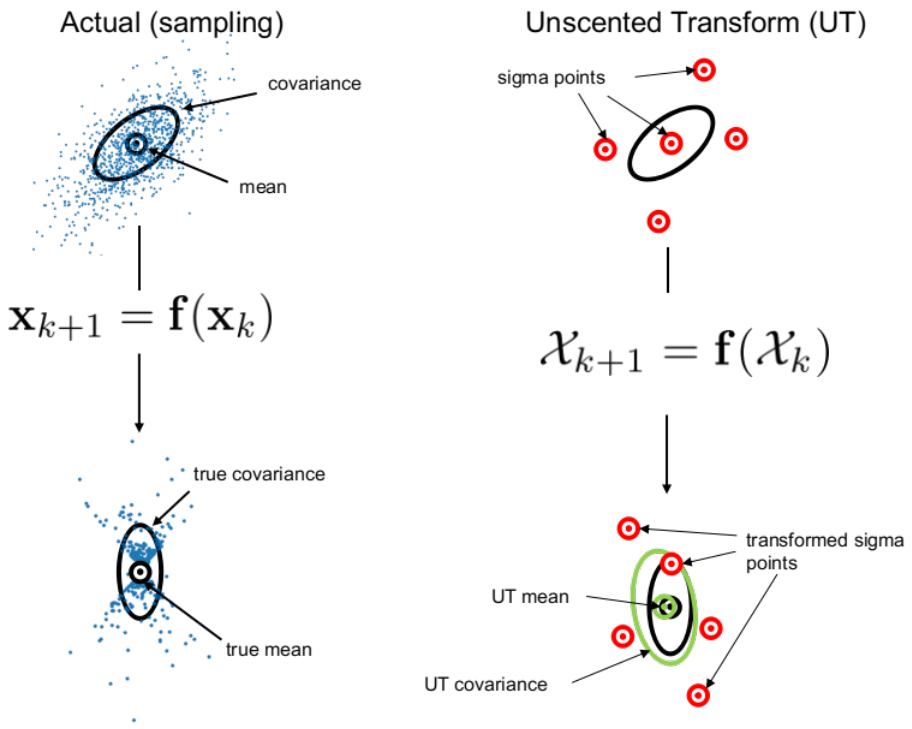


Figure 2.7: Unscented Transform

To formulate Unscented transform, first, let us consider a random variable $\mathbf{x}$ with mean $\bar{\mathbf{x}}$ and covariance $\mathbf{P}_{\mathbf{x}}$ having dimension L going through non-linear transformation $\mathbf{y}$ = $g(\mathbf{x})$. For this transformation, we need the statistics like mean and covariance of $\mathbf{y}$. To compute these statistics we first need the sigma points matrix $\mathbf{X}$ with dimension $2L+1$ for random variable $\mathbf{x}$ with sigma point vector $\mathbf{X}_j$ and corresponding weights W_j. Unscented transform is shown in Figure 2.7. Mathematically this procedure is given as:

$$
\begin{aligned}
\mathbf{X}_0 &= \bar{\mathbf{x}} \\
\mathbf{X}_j &= \bar{\mathbf{x}} + (\sqrt{(L+\gamma)\mathbf{P}_\mathbf{x}})_j \qquad 1 \leq j \leq L \\
\mathbf{X}_j &= \bar{\mathbf{x}} + (\sqrt{(L+\gamma)\mathbf{P}_\mathbf{x}})_j \qquad n+1 \leq j \leq 2L \\
W_0^{(m)} &= \frac{\gamma}{(L+\gamma)} \\
W_0^{(c)} &= \frac{\gamma}{(L+\gamma)} + (1-\alpha^2+\beta) \\
W_j^{(m)} &= W_j^{(c)} \\
&= \frac{1}{2(L+\gamma)} \qquad 1 \leq j \leq 2L
\end{aligned}
\tag{2.15}
$$

Where $\gamma = \alpha^2(L+\kappa) - L$ is a scaling parameter. α determines the distribution of the sigma points around mean $\bar{\mathbf{x}}$. κ is the additional scaling parameter which is normally set to zero. β is used to add the impact of prior statistics of the distribution of $\mathbf{x}$ and W_j are the corresponding weights of $\mathbf{X}_j$ that produces sigma points $\mathbf{X}_j$ with the same mean and covariance as the random variable $\mathbf{x}$. Subscript m or c in the weights signifies weights for mean or covariance respectively.

To compute mean $\bar{\mathbf{y}}$ and covariance $\mathbf{P}_\mathbf{y}$ of the nonlinear transformation, we first pass the sigma point vector $\mathbf{X}_j$ from the same nonlinear transformation $\mathbf{Y}_j = g(\mathbf{X}_j)$. Mathematically it is given as:

$$
\begin{aligned}
\mathbf{Y}_j &= g(\mathbf{X}_j) \qquad 0 \leq j \leq 2L \\
\bar{\mathbf{y}} &\approx \sum_{j=0}^{2L} W_j^m \mathbf{Y}_j \\
\mathbf{P}_y &\approx \sum_{j=0}^{2L} W_j^c [\mathbf{Y}_j - \bar{\mathbf{y}}][\mathbf{Y}_j - \bar{\mathbf{y}}]^T
\end{aligned}
\tag{2.16}
$$

where $\bar{\mathbf{y}}$ and $\mathbf{P}_\mathbf{y}$ are the approximate mean and covariance of the random variable $\mathbf{y}$ respectively.

Unscented Kalman Filter is the extension of the unscented transform in which the random variable $\mathbf{x}$ is modified in order to incorporate the process noise vector $\mathbf{v}$ and measurement noise vector $\mathbf{n}$. To do this first, the random vector at time instance $k = 0$ is chosen as $\mathbf{x}_0$ and its mean $\hat{\mathbf{x}}_0$ and covariance $\mathbf{P}_0$ are computed. Since the initial assumption is that process noise and measurement noise is zero-mean Gaussian noise their expected value is zero. Mathematically this algorithm is expressed as:

$$\begin{aligned}
\hat{\mathbf{x}}_0 &= E[\mathbf{x}_0] \\
\mathbf{P}_0 &= E[(\mathbf{x}_0 - \hat{\mathbf{x}}_0)(x_0 - \hat{\mathbf{x}}_0)^T] \\
\hat{\mathbf{x}}_0^a &= E[\mathbf{x}^a] \\
&= [\hat{\mathbf{x}}_0^T \quad 0 \quad 0]^T \\
\mathbf{P}_0^a &= E[(\mathbf{x}_0^a - \hat{\mathbf{x}}_0^a)(\mathbf{x}_0^a - \hat{\mathbf{x}}_0^a)^T] \\
&= \begin{bmatrix} \mathbf{P}_0 & 0 & 0 \\ 0 & \mathbf{P}_v & 0 \\ 0 & 0 & \mathbf{P}_n \end{bmatrix}
\end{aligned} \tag{2.17}$$

where $\hat{\mathbf{x}}_0^a$ and $\mathbf{P}_0^a$ are the mean and covariance of the modified random vector $\mathbf{x}^a$ respectively.

Further for $k \, \epsilon \{1, ..., \infty\}$ using sigma point computation scheme given in Equation 2.15 sigma points are computed for random vector $\hat{\mathbf{x}}^a$ as:

$$\mathbf{X}_{k-1}^a = [\hat{\mathbf{x}}_{k-1}^a \quad \hat{\mathbf{x}}_{k-1}^a \quad \pm \sqrt{(L+\gamma)\mathbf{P}_{k-1}^a}\,] \tag{2.18}$$

For a nonlinear transformation, $F[.]$ and measurement model $H[.]$ using the technique described in Equation 2.16 the estimated $a \; priori$ mean $\hat{\mathbf{x}}_k^-$ and covariance $\mathbf{P}_k^-$ on $\mathbf{x}_k$ are computed as:

$$
\begin{aligned}
\mathbf{X}^{\mathbf{x}}_{k|k-1} &= F[\mathbf{X}^{\mathbf{x}}_{k-1}, \mathbf{X}^{v}_{k-1}] \\
\hat{\mathbf{x}}^{-}_{k} &= \sum_{j=0}^{2L} W^{m}_{j} \mathbf{X}^{\mathbf{x}}_{j,k|k-1} \\
\mathbf{P}^{-}_{k} &= \sum_{j=0}^{2L} W^{c}_{j} [\mathbf{X}^{\mathbf{x}}_{j,k|k-1} - \hat{\mathbf{x}}^{-}_{k}][\mathbf{X}^{\mathbf{x}}_{j,k|k-1} - \hat{\mathbf{x}}^{-}_{k}]^{T} \\
\mathbf{Y}_{k|k-1} &= H[\mathbf{X}^{\mathbf{x}}_{k|k-1}, \mathbf{X}^{n}_{k-1}] \\
\hat{\mathbf{y}}^{-}_{k} &= \sum_{j=0}^{2L} W^{m}_{j} \mathbf{Y}_{j,k|k-1}
\end{aligned}
\tag{2.19}
$$

where $\mathbf{Y}_{k|k-1}$ is the sigma points of the transformed measurement noise vector at time step k and $\hat{\mathbf{y}}^{-}_{k}$ is the sampled mean of the measurement vector.

Finally, the estimated $a\ posteriori$ mean $\hat{\mathbf{x}}_k$ and covariance $\mathbf{P}_k$ after transformation is obtained as:

$$
\begin{aligned}
\mathbf{P}_{\hat{\mathbf{y}}_k \hat{\mathbf{y}}_k} &= \sum_{j=0}^{2L} W^{c}_{j} [\mathbf{Y}_{j,k|k-1} - \hat{\mathbf{y}}^{-}_{k}][\mathbf{Y}_{j,k|k-1} - \hat{\mathbf{y}}^{-}_{k}]^{T} \\
\mathbf{P}_{\mathbf{x}_k \mathbf{y}_k} &= \sum_{j=0}^{2L} W^{c}_{j} [\mathbf{X}_{j,k|k-1} - \hat{\mathbf{x}}^{-}_{k}][\mathbf{Y}_{j,k|k-1} - \hat{\mathbf{y}}^{-}_{k}]^{T} \\
K &= \mathbf{P}_{\hat{\mathbf{x}}_k \hat{\mathbf{y}}_k} \mathbf{P}^{-1}_{\hat{\mathbf{y}}_k \hat{\mathbf{y}}_k} \\
\hat{\mathbf{x}}_k &= \hat{\mathbf{x}}^{-}_{k} + K(\mathbf{y}_k - \hat{\mathbf{y}}^{-}_{k}) \\
\mathbf{P}_k &= \mathbf{P}^{-}_{k} - K \mathbf{P}_{\hat{\mathbf{y}}_k \hat{\mathbf{y}}_k} K^{T}
\end{aligned}
\tag{2.20}
$$

where K is the gain of the transformation.

For the next step $k+1$ mean and covariance obtained from Equation 2.20 for step k is used in Equation 2.18 and the whole process is repeated.

2.3 Markov Model

A Markov model is a probabilistic model for successive or temporal data. This model exploits the fact about the real world that given the present state, the future state of a system is independent of all past states, i.e., if the exact state of the system at present is known, knowledge of the past states to predict the future state is of no use as the information of all the past states of the system are encoded in the present state. Markov Model can be used to model an extremely large number of real-world sequential data like finances, language, weather, speech etc. The autoregressive model of Section 2.1.1 is a special case of a Markov model. Here, we specifically focus on discrete first-order Markov models, known as Markov chains.

For a sequential states $x_1, x_2, ..., x_N$ the joint probability distribution is given by

$$p(x_1, ..., x_N) = \prod_{k=1}^{N} p(x_k|x_1, ..., x_{k-1}) \tag{2.21}$$

For a state x_k dependent only on the most recent past state x_{k-1} the joint probability distribution of Equation 2.21 becomes the first order Markov chain and it is mathematically expressed as

$$p(x_1, ..., x_N) = p(x_1) \prod_{k=2}^{N} p(x_k|x_{k-1}) \tag{2.22}$$

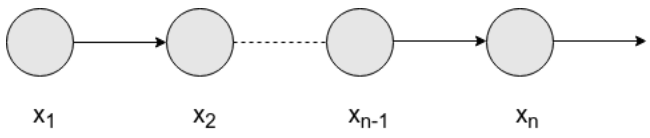


Figure 2.8: First order Markov chain of sequence $\{x_n\}$

The transition probability of a random variable from one state to another in a Markov

chain is given by

$$p_{ij} = Pr\{Z_n = j | Z_{n-1} = i\} \tag{2.23}$$

Where p_{ij} is the probability of transition of the state from state i to state j. Graphically, Markov chain is represented by the state transition diagram. It is a diagram showing the probability of a state transitioning in all other possible states or remaining in the same current state. Probabilities of state transition are collectively represented by a transition matrix. The transition matrix is a matrix containing probabilities p_{ij}. For a state, a sum of all the transition probabilities is 1.

For a system consisting of three possible states $k = 1, 2, 3$ the transition matrix is given by

$$\boldsymbol{T} = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix}$$

where p_{ij} is the probability of the system transitioning from state i to state j.

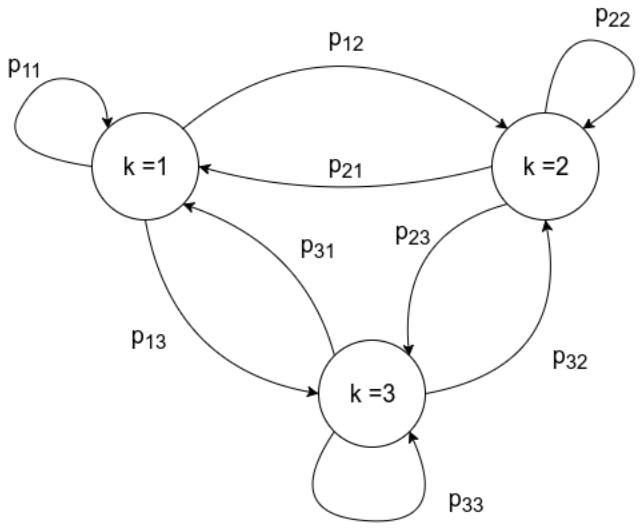


Figure 2.9: State transition diagram

For previous probability A_{n-1} the new state is given by A_n and expressed as

$$A_n = \boldsymbol{T}\, A_{n-1} \tag{2.24}$$

2.4 Gaussian Mixture Model

The Gaussian distribution is a distribution following a characteristic bell curve. For a Gaussian random vector $\mathbf{X} = [x_1, x_2, ..., x_M]$ with dimension M, mean $\mu \in R_{Mx1}$ and covariance $\Sigma \in S_{MxM}$ the probability of the Gaussian distribution is given by

$$P(x) = \frac{1}{(2\pi)^{M/2}\,|\Sigma|^{1/2}}\, \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right) \tag{2.25}$$

Gaussian mixture model states that for M multivariate Gaussian distributions, the probability of the mixture of all the Gaussian is

$$p(x) = \sum_{i=1}^{M} w_i\, \mathcal{N}(x|\mu_i, \Sigma) \tag{2.26}$$

Here w_i is the weight i.e., the probability of the prior of i^{th} distribution. The sum of all the w_i equals to one.

For two-dimensional random variables with mean m_0 and m_1, covariances C_0 and C_1 and probabilities $p(x_0) = w_0$ and $p(x_1) = w_1$ and the Gaussian mixture mean μ_i is given by

$$\mu_i = w_0 m_0 + w_1 m_1 \tag{2.27}$$

And the covariance matrix is given by

$$\Sigma_i = \underbrace{w_0 \left(C_0 + m_0 m_0^T\right) + w_1 \left(C_1 + m_1 m_1^T\right)}_{E[x_i x_i^T] = 2^{nd}\, moment} - \mu_i \mu_i^T \tag{2.28}$$

Switching Neural Network Tracker

In this chapter, we discuss and formulate an algorithm to simultaneously track the states and modes of a dynamical system. Separate neural networks are trained for each mode and used as state transition functions, while the Unscented transform and Gaussian-mixture models are used to capture nonlinear effects and estimate the state covariance and mean to give final moments of the system to be used for next cycle of operation. This algorithm is given the name Switching Neural Network Tracker (SNNT).

3.1 Model

The model we use in this thesis is given by a probability graph diagram shown in Figure 3.1. This diagram consists of the distinct entities: modes, states, transition functions, and measurements. The upper portion of the graph gives the modes of the system, which follow a Markov chain rule, and the lower portion of the graph gives the hidden states which are represented by state space model. The hidden states of the system are quantified by the observation, also known as measurements of the states.

At any time step k, mode z_k is descried by the probability of the system being in mode z_k conditioned on previous mode z_{k-1} and the probability of the system being in state x_k conditioned on previous state x_{k-1}. Further, state x_k depends on probability of state x_k conditioned on previous state x_{k-1} and current mode z_k.

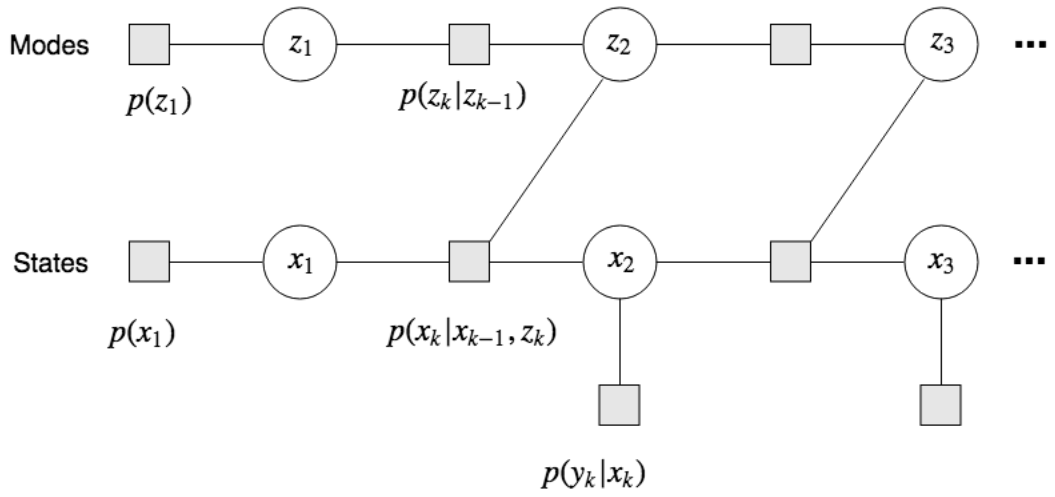


Figure 3.1: Factor graph of the nonlinear switching dynamical system model.

3.1.1 Approximation of transition functions

"Function" is a technical term used to signify mappings between input and output. When the input and the mapping are known, the output can be easily quantified. As the nonlinearity of the input-output mapping increases, it becomes difficult to find the true relationship between input and output, but this relationship can be approximated. The technical term for it is function approximation. Function approximation is the fundamental objective in a large number of real-world tasks like regression, data classification, and pattern recognition. For a system with the input vector X and output vector Y, the mapping $X \mapsto Y$ is given by the function $F(\cdot)$. The approximation of the mapping $F(\cdot)$ is given by $\hat{F}(\cdot)$ such that $Y \approx \hat{F}(X)$.

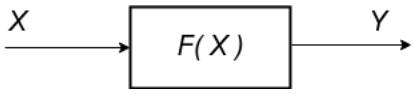


Figure 3.2: Input-Output mapping function

There are various techniques that can be used to approximate the input-output relationship, that is, function, of a system. Some of the most widely used function approximation

methods are MLP Neural Networks[15] ad RBF Neural Networks[12]. An RBF Neural Network is a neural network in which there is a single hidden layer with multiple nodes. Each node's activation function is a radial bias function. In this thesis, MLP Neural Networks are used to approximate the update function due to its wide use and simplicity of application.

Neural networks are some of the most pervasive technology that are currently being used. A three-layer neural network is shown in Figure 3.3. With weights in each hidden layers being $\mathbf{W}_i$ for $i = 1, 2, 3$, activation function $\mathbf{\Gamma}[\cdot]$, and input feature vector $\boldsymbol{X}$, the input-output relationship can be expressed as

$$\mathbf{y} = \mathbf{\Gamma}[\boldsymbol{W}_3\mathbf{\Gamma}[\boldsymbol{W}_2[\mathbf{\Gamma}[\boldsymbol{W}_1\mathbf{X}]]] \tag{3.1}$$

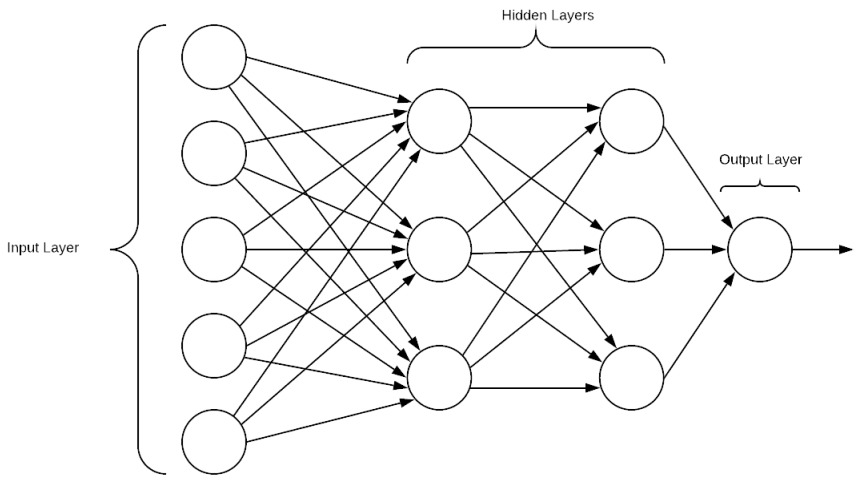


Figure 3.3: A three-layer neural network

In this thesis Keras[5], a python library, is used to create neural network models and a learning rate of 0.001 is used for all dataset to create approximate functions.

3.1.2 Spatial feature extraction

The model and tracking system developed in this thesis is compatible with any form of state data, however we consider spatial tracking as a special case. Objects are considered to be moving in a spatial coordinate system and those objects produce (x, y) coordinates in 2D space with modes label '$\mathbf{z}$' with the passage of time. Features for training the neural networks are generated as follows: For a two-mode system with $N_0 + N_1$ available data points, with N_0 data points in mode $z = 0$, and N_1 data points in mode $z = 1$, the dataset matrices are given by

$$DS_0 = \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \vdots & \vdots \\ x_{N_0} & y_{N_0} \end{bmatrix} \qquad DS_1 = \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \vdots & \vdots \\ x_{N_1} & y_{N_1} \end{bmatrix}.$$

The feature space is generated using a specified number of history points R and the next value in the data-set as a target value. For a three history points feature-space, the generated features and targets are given by

$$\mathbf{X}_0 = \begin{bmatrix} x_3 & y_3 & x_2 & y_2 & x_1 & y_1 \\ x_4 & y_4 & x_3 & y_3 & x_2 & y_2 \\ x_5 & y_5 & x_4 & y_4 & x_3 & y_3 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{N_0-1} & y_{N_0-1} & x_{N_0-2} & y_{N_0-2} & x_{N_0-3} & y_{N_0-3} \end{bmatrix}, \ \mathbf{y}_0 = \begin{bmatrix} x_4 & y_4 \\ x_5 & y_5 \\ x_6 & y_6 \\ \vdots & \vdots \\ x_{N_0} & y_{N_0} \end{bmatrix}$$

$$\mathbf{X}_1 = \begin{bmatrix} x_3 & y_3 & x_2 & y_2 & x_1 & y_1 \\ x_4 & y_4 & x_3 & y_3 & x_2 & y_2 \\ x_5 & y_5 & x_4 & y_4 & x_3 & y_3 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{N_1-1} & y_{N_1-1} & x_{N_1-2} & y_{N_1-2} & x_{N_1-3} & y_{N_1-3} \end{bmatrix}, \mathbf{y}_1 = \begin{bmatrix} x_4 & y_4 \\ x_5 & y_5 \\ x_6 & y_6 \\ \vdots & \vdots \\ x_{N_1} & y_{N_1} \end{bmatrix},$$

where $\mathbf{X}_0$ and $\mathbf{X}_1$ are the feature matrices, and $\mathbf{y}_0$ and $\mathbf{y}_1$ are the target value matrices for modes $z = 0$ and $z = 1$, respectively.

Thus the generated data sets are now ready to be used for training the neural networks. For each mode $z = 0, 1$, we use the training model given in the Figure 3.4. After training is complete, two approximation function models: $\hat{F}_0$ and $\hat{F}_1$ are obtained for mode $z = 0$ and $z = 1$, respectively.

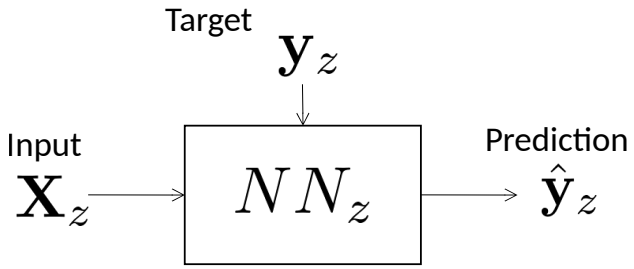


Figure 3.4: A neural network training model for mode z with training data $\mathbf{X}_z$ and target value $\mathbf{y}_z$.

3.1.3 Modes

The switching systems can only transition to the specific dynamics from a set of dynamics called modes. Modes are the discrete step that the dynamical switching system transitions to during the switching process. The dynamical switching system is free to switch to different modes based on it's system dynamics and statistical characteristics. Modes are represented numerically by numbering systems like 0, 1, and so on. This switching process is approximated by Markovian dynamics as given in Section 2.3 with transition matrix

$\boldsymbol{T}$. For a two-mode system with modes numbering $z_k = 0$ and $z_k = 1$, the probability of switching is described as

$$p(z_k = 0|z_{k-1} = 0) = P_{00},\ p(z_k = 1|z_{k-1} = 0) = P_{10}$$
$$p(z_k = 0|z_{k-1} = 1) = P_{01},\ p(z_k = 1|z_{k-1} = 1) = P_{11},$$

where P_{00} is the probability that the system in mode 0 stays in mode 0; P_{01} is the probability that the system in mode 1 switches to mode 0. P_{11} is the probability that system in mode 1 stays in mode 1, and P_{10} is the probability that the system in mode 0 switches to mode 1.

These probabilities can be represented in matrix form, and it is commonly known as the transition matrix of the Markov model which is given by

$$\boldsymbol{T} = \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}. \tag{3.2}$$

3.1.4 States

States are the hidden continuous-valued vectors that the system attains with the passage of time. States can be any quantity like velocity, coordinates, or other spatial characteristics of the system. Generally state vectors are denoted by $\mathbf{x}_k$, where $k = 1, 2, 3, ...$ As noted above, our examples use spatial co-ordinates as states of the system with each 2D point being represented as $\mathbf{p}_k$. The first position in $\mathbf{x}_k$ is occupied by the k^{th} point i.e., $\mathbf{p}_k$, the second position is occupied by $(k-1)^{th}$ points and so on up to R history points. For a system with R history points, there are $2R$ entries in the state vector. In matrix form, it is

expressed as

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{p}_k \\ \mathbf{p}_{k-1} \\ \vdots \\ \mathbf{p}_{k-R} \end{bmatrix}, \tag{3.3}$$

where an individual point $\mathbf{p}_k$ is expressed using the x-coordinate (r_k^x) and y-coordiante (r_k^y) at time k

$$\mathbf{p}_k = \begin{bmatrix} r_k^x \\ r_k^y \end{bmatrix}. \tag{3.4}$$

The current state $\mathbf{x}_k$ depends on the previous state $\mathbf{x}_{k-1}$ and the current mode z_k of the system. The nonlinear dynamical system is modeled as

$$\mathbf{x}_k | \mathbf{x}_{k-1}, z_k = f_{z_k}(\mathbf{x}_{k-1}) + \mathbf{v}_k, \tag{3.5}$$

where $f_{z_k}(\mathbf{x}_{k-1})$ is a mode-dependent transition function learned using the neural network on the training dataset for mode z_k, and $\mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \sigma_p^2 \mathbf{I})$ is residual process noise of the

system. In matrix form, the states are represented as

$$\mathbf{x}_k|\mathbf{x}_{k-1}, z_k = \begin{bmatrix} f_{z_k}(\mathbf{x}_{k-1}) + \mathbf{v}_k \\ \mathbf{p}_{k-1} \\ \vdots \\ \mathbf{p}_{k-R} \end{bmatrix} \tag{3.6}$$

$$= \begin{bmatrix} f_{z_k}(\mathbf{x}_{k-1}) \\ \mathbf{p}_{k-1} \\ \vdots \\ \mathbf{p}_{k-R} \end{bmatrix} + \mathbf{B}\mathbf{v}_k, \tag{3.7}$$

where matrix $\mathbf{B}$ is a matrix that applies residual noise $\mathbf{v}_k \in \mathbb{R}^2$ only to the new components of the states and has the form

$$\mathbf{B} = \begin{bmatrix} \mathbf{I}_{2\times 2} \\ \mathbf{0}_{2\times 2} \\ \vdots \\ \mathbf{0}_{2\times 2} \end{bmatrix}. \tag{3.8}$$

Since the first two elements of $\mathbf{x}_k|\mathbf{x}_{k-1}, z_k$ are the only random variables and rest of the elements are just the delayed versions, the transition probability $p(\mathbf{x}_k|\mathbf{x}_{k-1}, z_k)$ can then be represented as

$$p(\mathbf{x}_k|\mathbf{x}_{k-1}, z_k) = \mathcal{N}(\mathbf{p}_k; f_{z_k}(\mathbf{x}_{k-1}), \sigma_p^2\mathbf{I}_2). \tag{3.9}$$

3.1.5 Measurements

The states of the system are hidden, and measurements provide the only indirect information about the current state. Measurement $\mathbf{y}_k$ is modeled as the first entry $\mathbf{p}_k$ in the state $\mathbf{x}_k$

corrupted by noise with distribution $\mathcal{N}(0, \sigma_n^2 \mathbf{I}_2)$. That is,

$$\mathbf{y}_k = \mathbf{H}\mathbf{x}_k + \mathbf{n}_k = \begin{bmatrix} 1 & 0 & 0 & 0 \dots \\ 0 & 1 & 0 & 0 \dots \end{bmatrix} \begin{bmatrix} r_k^x \\ r_k^y \\ r_{k-1}^x \\ r_{k-1}^y \\ \vdots \end{bmatrix} + \mathbf{n}_k, \tag{3.10}$$

where $\mathbf{H}$ is the observation matrix that realizes noiseless measurements from the states. Therefore, the likelihood of the measurement $\mathbf{y}_k$ given the state $\mathbf{x}_k$ is given by

$$p(\mathbf{y}_k|\mathbf{x}_k) = \mathcal{N}(\mathbf{y}_k; \mathbf{H}\mathbf{x}_k, \sigma_n^2 \mathbf{I}_2). \tag{3.11}$$

3.2 Tracking Algorithm

Modes and states of the dynamical system need to be tracked in order to successfully understand and estimate system dynamics. The posterior probability $p(z_k|\boldsymbol{y}_k)$ of the system being in mode z_k must be computed at each time step k along with the state posteriors $p(\boldsymbol{x}_k|\boldsymbol{y}_k)$. The evaluation of posterior modes and states are considered, in turn, in the following sections.

3.2.1 Modes

Modes z_k are the distinct and discrete property that the switching dynamical system can switch to over time, k. To track modes of the system, we need to be able to find the probability associated with the mode conditioned on the measurement at any instant of time. For time instant k, with mode z_k, state vector $\mathbf{x}_k$, and measurement vector $\mathbf{y}_k$, we need to find $p(z_k|\mathbf{y}_k)$ to successfully characterize the posterior of z_k. To compute the posterior, we employ the assumption that the Modes follow a Markov chain (see Section 3.1.3) and treat $p(z_k|\mathbf{y}_{k-1}, ..., \mathbf{y}_1)$ as the prior of z_k at time k. From the graphical model given in Figure 3.1,

at time k we have

$$p(z_k, \mathbf{x}_k, \mathbf{x}_{k-1}, \mathbf{y}_k) = p(z_k|\boldsymbol{y}_{k-1})\, p(\mathbf{x}_k|\mathbf{x}_{k-1}, z_k)\, p(\mathbf{x}_{k-1}|\mathbf{y}_{k-1})\, p(\mathbf{y}_k|\mathbf{x}_k). \tag{3.12}$$

Conditioning on $\mathbf{y}_k$ and marginalizing out the state variables, we obtain

$$p(z_k|\mathbf{y}_k) \propto p(z_k|\boldsymbol{y}_{k-1}) \int\int p(\mathbf{x}_k|\mathbf{x}_{k-1}, z_k)\, p(\mathbf{x}_{k-1}|\mathbf{y}_{k-1})\, p(\mathbf{y}_k|\mathbf{x}_k)\, d\mathbf{x}_{k-1}\, d\mathbf{x}_k. \tag{3.13}$$

As shown below, the previous state distribution $p(\mathbf{x}_{k-1}|\mathbf{y}_{k-1})$ is approximated as a Gaussian distribution; thus, we have

$$p(\mathbf{x}_{k-1}|\mathbf{y}_{k-1}) \approx \mathcal{N}(\mathbf{x}_{k-1}; \boldsymbol{\mu}_{k-1}, \boldsymbol{\Sigma}_{k-1}). \tag{3.14}$$

Evaluating Equation 3.13 requires two multi-dimensional integrations, and it is impractical to solve this integration using conventional mathematical techniques. However, the integrations may be approximated using UT described in Section 2.2.2. To approximate the probability distribution resulting from the integrations, the following steps are used.

First, the state vector is augmented to incorporate process and measurement noise as

$$\mathbf{x}^a = [\mathbf{x}_{k-1}, \mathbf{v}_k, \mathbf{n}_k]^T. \tag{3.15}$$

Since the process and measurement noises are assumed to have zero mean, the mean $\bar{\mathbf{x}}^a$ and covariance $\mathbf{C}^a$ of the augmented vector $\mathbf{x}^a$ are given as

$$\bar{\mathbf{x}}^a = [\mu_{k-1}, \mathbf{0}, \mathbf{0}]^T, \quad \mathbf{C}^a = \begin{bmatrix} \boldsymbol{\Sigma}_{k-1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma}_p & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{\Sigma}_n \end{bmatrix}, \tag{3.16}$$

where $\boldsymbol{\Sigma}_p = \sigma_p^2\mathbf{I}$ and $\boldsymbol{\Sigma}_n = \sigma_n^2\mathbf{I}$ represent the covariance matrices of the process noise and measurement noise, respectively. The length of the augmented state vector $\bar{\mathbf{x}}^a$ is $L =$

$n_{state} + n_{process} + n_{meas}$, where $n_{state} = 2R$ (recall R is the number points in the history buffer, so for R history points there are $2R$ coordinates), $n_{process}$ is the dimension of the process noise which is 2 for the most recent (x, y) in the state vector, and n_{meas} is the dimension of the measurement noise which is also 2. Using Equations 3.16 and 2.15, weights $W_0^{(m)}$, $W_0^{(c)}$, $W_j^{(m)}$ and $W_j^{(c)}$ are computed, and sigma points $\mathbf{S}$ are generated.

To perform the UT, Equation 3.13 is reorganized as

$$p(z_k|\mathbf{y}_k) \propto p(z_k|\boldsymbol{y}_{k-1}) \int \underbrace{\int p(\mathbf{x}_k|\mathbf{x}_{k-1}, z_k)\, p(\mathbf{x}_{k-1}|\mathbf{y}_{k-1})\, d\mathbf{x}_{k-1}}_{integral1 \to q(\mathbf{x}_k|z_k)} p(\mathbf{y}_k|\mathbf{x}_k)\, d\mathbf{x}_k \tag{3.17}$$

To approximate $q(\mathbf{x}_k|z_k)$ resulting from $integral1$ in Equation 3.17, we hold z_k fixed and propagate sigma points $\mathbf{S}$ through the system described by Equation 3.5. Using the new sigma points, denoted $\mathbf{S}_{\mathbf{x}_k}$, to represent the distribution $q(\mathbf{x}_k|z_k)$ over $\boldsymbol{x}_k$, we then have

$$p(z_k|\mathbf{y}_k) \propto p(z_k|\boldsymbol{y}_{k-1}) \underbrace{\int q(\mathbf{x}_k|z_k)\, p(\mathbf{y}_k|\mathbf{x}_k)\, d\mathbf{x}_k}_{integral2 \to L(\mathbf{y}_k|z_k)} \tag{3.18}$$

Using sigma points $\mathbf{S}_{\mathbf{x}_k}$ and the likelihood function given in Equation 3.11, $integral2$ in Equation 3.18 is approximated. This results in new sigma points $\mathbf{S}_{\mathbf{y}_k}$ representing the marginalized likelihood $L(\mathbf{y}_k|z_k)$.

Sigma points $\mathbf{S}_{\mathbf{y}_k}$ are used to approximate $\boldsymbol{\mu}_{\mathbf{y}_k}$ and $\mathbf{C}_{\mathbf{y}_k}$, the mean and covariance of the marginalized likelihood. Adopting a Gaussian approximation, we then have

$$L(\mathbf{y}_k|z_k) \approx \mathcal{N}(\mathbf{y}_k; \mu_{\mathbf{y}_k}, \mathbf{C}_{\mathbf{y}_k}). \tag{3.19}$$

Finally, using the measurement data $\mathbf{y}_k = \mathbf{y}_k^{meas}$, the marginalized likelihood 3.19 can be evaluated and combined with the prior $p(z_k|\boldsymbol{y}_{k-1})$ to produce a posterior

$$p(z_k|\mathbf{y}_k) \propto p(z_k|\boldsymbol{y}_{k-1})\, L(\mathbf{y}_k|z_k) \tag{3.20}$$

for z_k. This technique is repeated for each mode, that is for $z_k = 0$ and $z_k = 1$, which gives

$$\begin{aligned} p(z_k = 0|\mathbf{y}_k = \mathbf{y}_k^{meas}) &\propto p(z_k = 0|\boldsymbol{y}_{k-1})\, L(\mathbf{y}_k = \mathbf{y}_k^{meas}|z_k = 0) \\ p(z_k = 1|\mathbf{y}_k = \mathbf{y}_k^{meas}) &\propto p(z_k = 1|\boldsymbol{y}_{k-1})\, L(\mathbf{y}_k = \mathbf{y}_k^{meas}|z_k = 1). \end{aligned} \tag{3.21}$$

In practice, the likelihoods from 3.19 are very small, and the normalization required of the proportionality in 3.21 should be computed in the log-domain.

Once $p(z_k|\mathbf{y}_k = \mathbf{y}_k^{meas})$ is known, the predicted probability of the mode of the system at time instant $k+1$, that is $p(z_{k+1}|\mathbf{y}_k = \mathbf{y}_k^{meas})$, is computed using the probability entries of the Markov transition matrix $\boldsymbol{T}$

$$p(z_{k+1}|\mathbf{y}_k = \mathbf{y}_k^{means}) = \sum_{z_k \epsilon \{0,1\}} p(z_{k+1}|z_k)\, p(z_k|\mathbf{y}_k = \mathbf{y}_k^{means}). \tag{3.22}$$

After this step, the conditioned probability $p(z_{k+1}|\mathbf{y}_k = \mathbf{y}_k^{meas})$ is treated as prior on the state z_{k+1} for time step $k+1$, and denoted simply $p(z_{k+1}|\boldsymbol{y}_k)$. Equation 3.22 can be computed in vector form as

$$\texttt{p\_zkplus1} = \boldsymbol{T}\, \texttt{p\_zk\_post}, \tag{3.23}$$

where $\boldsymbol{T}$ is the transition matrix of the Markov model described in Section 3.1.3 and

$$\texttt{p\_zkplus1} = \begin{bmatrix} p(z_{k+1} = 0|\boldsymbol{y}_k) \\ p(z_{k+1} = 1|\boldsymbol{y}_k) \end{bmatrix} \tag{3.24}$$

are the predicted mode probabilities at time $k+1$, and

$$\texttt{p\_zk\_post} = \begin{bmatrix} p(z_k = 0|\mathbf{y}_k) \\ p(z_k = 1|\mathbf{y}_k) \end{bmatrix} \tag{3.25}$$

are the posterior probabilities at time k.

3.2.2 States

We now turn to estimating the hidden states of the dynamical system. To propagate the state uncertainty, we start with the posterior from the previous state, that is $p(\mathbf{x}_{k-1}|\mathbf{y}_{k-1})$. Then the current posterior of $\mathbf{x}_k$ is computed as

$$\begin{aligned} p(\mathbf{x}_k|\mathbf{y}_k) &\propto \sum_{z_k \epsilon \{0,1\}} p(z_k|\boldsymbol{y}_k) \int p(\mathbf{x}_k|\mathbf{x}_{k-1}, z_k)\, p(\mathbf{x}_{k-1}|\mathbf{y}_{k-1})\, p(\mathbf{y}_k|\mathbf{x}_k) d\mathbf{x}_{k-1} \\ &= p(z_k = 0|\boldsymbol{y}_k) \underbrace{\int p(\mathbf{x}_k|\mathbf{x}_{k-1}, z_k = 0)\, p(\mathbf{x}_{k-1}|\mathbf{y}_{k-1}) d\mathbf{x}_{k-1}\, p(\mathbf{y}_k|\mathbf{x}_k)}_{UKF1 \rightarrow \mathcal{N}(\mathbf{x}_k; \boldsymbol{m}_0, \boldsymbol{C}_0)}. \\ &+ p(z_k = 1|\boldsymbol{y}_k) \underbrace{\int p(\mathbf{x}_k|\mathbf{x}_{k-1}, z_k = 1)\, p(\mathbf{x}_{k-1}|\mathbf{y}_{k-1}) d\mathbf{x}_{k-1}\, p(\mathbf{y}_k|\mathbf{x}_k)}_{UKF2 \rightarrow \mathcal{N}(\mathbf{x}_k; \boldsymbol{m}_1, \boldsymbol{C}_1)} \end{aligned} \tag{3.26}$$

The bracketed quantities above represent typical non-linear filtering updates mapping a state prior, likelihood, and transition to a state posterior. For these updates, we employ the UKF for each mode. For mode $z_k = 0$ the UKF process is UKF_1, and for mode $z_k = 1$ the UKF process is UKF_2. Since the output of the UKF is a Gaussian [26], the above equation can be represented as a Gaussian mixture

$$p(\mathbf{x}_k|\mathbf{y}_k) = \underbrace{p(z_k = 0|\boldsymbol{y}_k)}_{w_0} \mathcal{N}(\mathbf{x}_k; \boldsymbol{m}_0, \boldsymbol{C}_0) + \underbrace{p(z_k = 1|\boldsymbol{y}_k)}_{w_1} \mathcal{N}(\mathbf{x}_k; \boldsymbol{m}_1, \boldsymbol{C}_1). \tag{3.27}$$

The weights, w_0 and w_1, are computed from the mode posteriors as derived in Section 3.2.1.

To maintain simplicity in the posterior state representation, we use a single Gaussian approximation

$$p(\mathbf{x}_k|\mathbf{y}_k) \approx \mathcal{N}(\mathbf{x}_k; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k). \tag{3.28}$$

The mean and covariance of the posterior $p(\mathbf{x}_k|\mathbf{y}_k)$ can then be computed using the mixture model described in Section 2.4 using $w_0, \boldsymbol{m}_0, \boldsymbol{C}_0$ as statistics of the first Gaussian and

$w_1, \boldsymbol{m}_1, \boldsymbol{C}_1$ as statistics of second Gaussian. Equations 2.26, 2.27 and 2.28 provide

$$\boldsymbol{\mu}_k = w_0 \boldsymbol{m}_0 + w_1 \boldsymbol{m}_1$$

$$\boldsymbol{\Sigma}_k = \underbrace{w_0 \left(\boldsymbol{C}_0 + \boldsymbol{m}_0 \boldsymbol{m}_0^T\right) + w_1 \left(\boldsymbol{C}_1 + \boldsymbol{m}_1 \boldsymbol{m}_1^T\right)}_{E[\boldsymbol{x}_k \boldsymbol{x}_k^T] = 2^{nd}\, moment} - \boldsymbol{\mu}_k \boldsymbol{\mu}_k^T$$

where $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ are, respectively, the mean vector and the covariance matrix of the mixture 3.27 and approximation 3.28. Thus, so computed $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ becomes the new statistics in Equation 3.16 for next iteration at time $k + 1$.

Results and Analysis

In this chapter, we present the applications of the switching algorithm that is Switching Neural Networks Tracker (SNNT), developed in Chapter 3 to real and synthetic data sets.

4.1 Honeybee Data

4.1.1 Dataset description

A honeybee colony has bees with different roles, including worker bees responsible for gathering nectar for the colony. To communicate the location of nectar source to other bees, a worker bee performs executes pattern of movements known as honeybee dance [6]. Bees uses the sun as a reference point and communicate the angle of the source of food with a dance pattern. The waggle dance is performed to signal the distance of source of food from the colony, and the dancing bee returns to the initial state using the circular left or right motion. A visual representation of a honeybee performing the signature dance pattern to communicate source of food is shown in Figure 4.1. Honeybee dataset [23] used in this thesis was compiled by a group of researchers at the Georgia Institute of Technology. This dataset contains six videos capturing the motion of different bees in a bee comb and six corresponding sequences of the dataset named $Sequence\ 1$ through $Sequence\ 6$. A frame of video from the $Sequence\ 6$ dataset is shown in Figure 4.2. Each sequence of dataset contains the spatial coordinates of one of the tracked bees performing the dance and contains labels for that sequence. The labels at each point in time are those of the three

movement patterns: left turn, waggle, and right turn. In this work we combined the left turn and right turn motions classes into a single "turn" category. This gives us two modes within the switching system: turn and waggle.

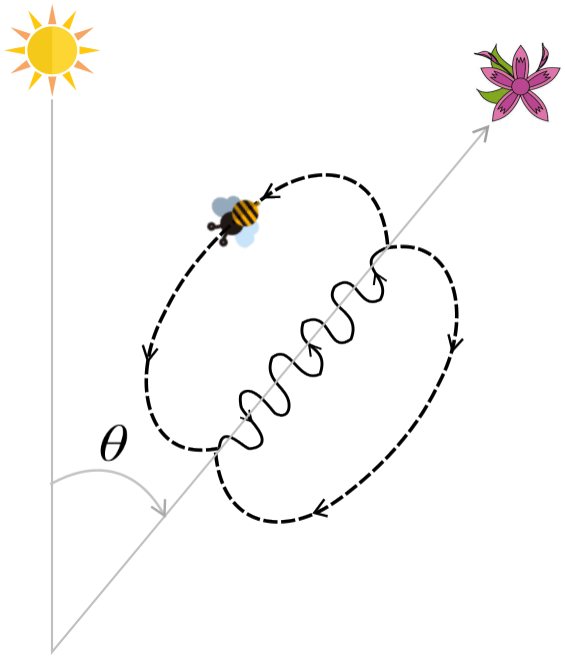


Figure 4.1: A visual representation of a honeybee dance pattern along with food source and a reference point. The angle and duration of the waggle indicate the angle and distance, respectively, to the food source.

4.1.2 Training details

We trained one neural network to approximate the waggle dynamics and another for the turning dynamics. To train the neural networks the following procedure was used: first, all mean shifted data points from $Sequence\ 3$ are taken and fractions of data containing single motion patterns are separated, for example, this dataset contains 603 data points from that 16 data chunks are taken containing different motions like turn and waggle. To increase the size of dataset each data chunk is rotated through random angles to create more data chunks. Using the method described in Section 3.1.2 features are generated for history

Figure 4.2: A frame of a video from $Sequence\ 6$ dataset capturing the dance pattern of a bee with the spatial path followed by the dancing honeybee superimposed on the frame.

$R = 7$ on these separate data chunks.

The process described above is repeated for $Sequence\ 4$ and $Sequence\ 5$ and concatenating turn and waggle set in two different arrays we finally obtain the training sets $\mathbf{X}_{turn}$, $\mathbf{y}_{turn}$ and $\mathbf{X}_{waggle}$, $\mathbf{y}_{waggle}$ for the turn and waggle motion respectively. Mode $z_k = 0$ is assigned for the turn and mode $z_k = 1$ is assigned for the waggle. After training two separate neural networks as described in Section 3.1.1 we obtain two functions $\hat{F}_{z=0}$ and $\hat{F}_{z=1}$ for the turn and waggle models respectively.

To minimize the abrupt switching between states in the Markov chain, the transition matrix is carefully selected to make the probability of transition to different states low. We use transition matrix given in 4.1 with the first row begin transition probabilities for turn mode and second row for waggle mode.

$$\boldsymbol{T} = \begin{bmatrix} 0.9999 & 0.0001 \\ 0.0001 & 0.9999 \end{bmatrix} \tag{4.1}$$

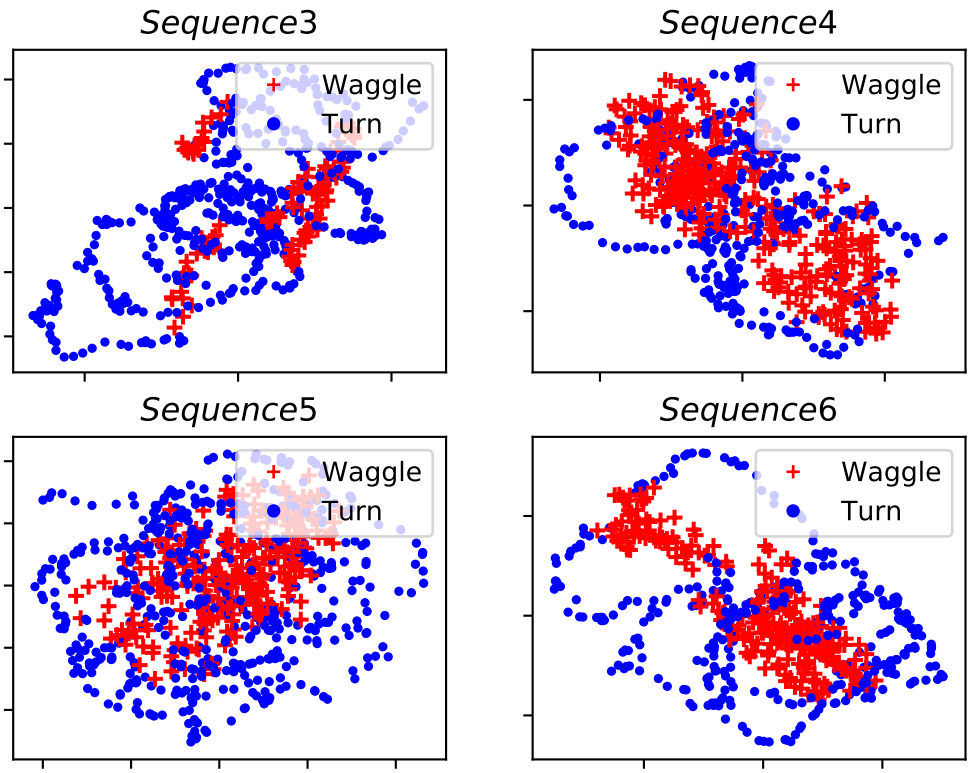


Figure 4.3: An illustration of different motions patterns in Honeybee dance dataset starting from top left $Sequence\ 3$, $Sequence\ 4$, $Sequence\ 5$ and $Sequence\ 6$.

The process noise for each mode was computed as follows: for waggle mode the training data $\mathbf{X}_{waggle}$ was passed to the trained neural network $\hat{F}_{waggle}$ and the residual noise was computed using $\hat{\mathbf{y}}_{waggle}$ and $\mathbf{y}_{waggle}$ after that, the residual noise was used to find the error covariance, giving the process noise for neural network model for waggle mode. This process is repeated for turn mode and process noise for turn neural network model was also obtained. Measurement noise, $\sigma_n = 0.4$ was selected for both SNNT and KF algorithms.

Optimal Kalman filter: Below, we use an optimal Kalman filter as a point of comparison to judge the state tracking performance of our algorithm. In order to make the comparison fair, we train for the optimal transition matrix using Adam optimizer and use the

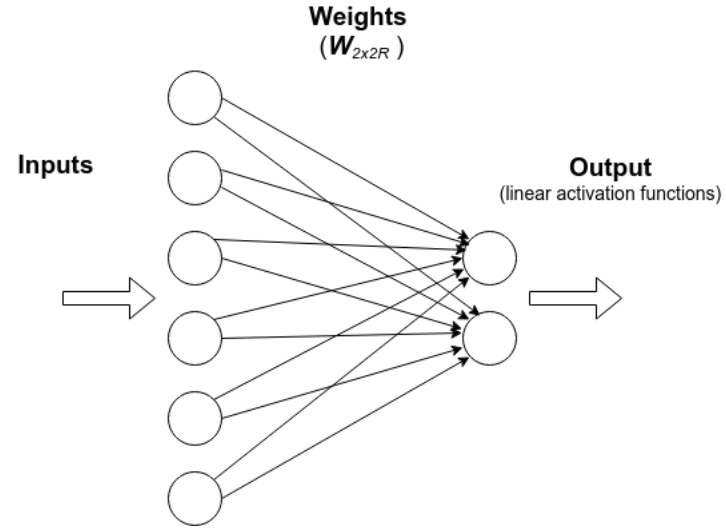


Figure 4.4: An architecture of feed-forward neural network used to train the first two rows of the transition matrix of a Kalman filter.

same training data that was used in the learning phase of SNNT. We use a neural network framework in an intuitive way to obtain the transition matrix as follows: For a six-state system the Kalman filter transition matrix has a size of 6×6, and the first two rows of the transition matrix gives us the prediction of next point given the historical points and last four rows simply copy the previous states to current state. So the first two rows of the transition matrix are obtained by training a feed-forward neural network with architecture given in Figure 4.4. A neural network with input and output with no hidden layer and linear activation function is trained on training data for 200 epoches until the training losses saturate to learn a $2 \times 2R$ matrix. This matrix is essentially a mapping that predicts next point in the robot path using R history points. This matrix is then assigned as first two rows of the transition matrix for the optimally learned KF.

4.1.3 Results

Here we use data points from $Sequence\,6$ with 609 samples of data to evaluate our algorithm. We fix the time history, $R = 7$ and start predicting the turn and waggle probabilities

with increasing time k. The sum of waggle and turn probabilities is 1 so whenever turn probability becomes greater than the waggle probability the the system dynamics switches to turn mode and vice versa. Mode estimation and probabilities of individual modes are given in the figure below.

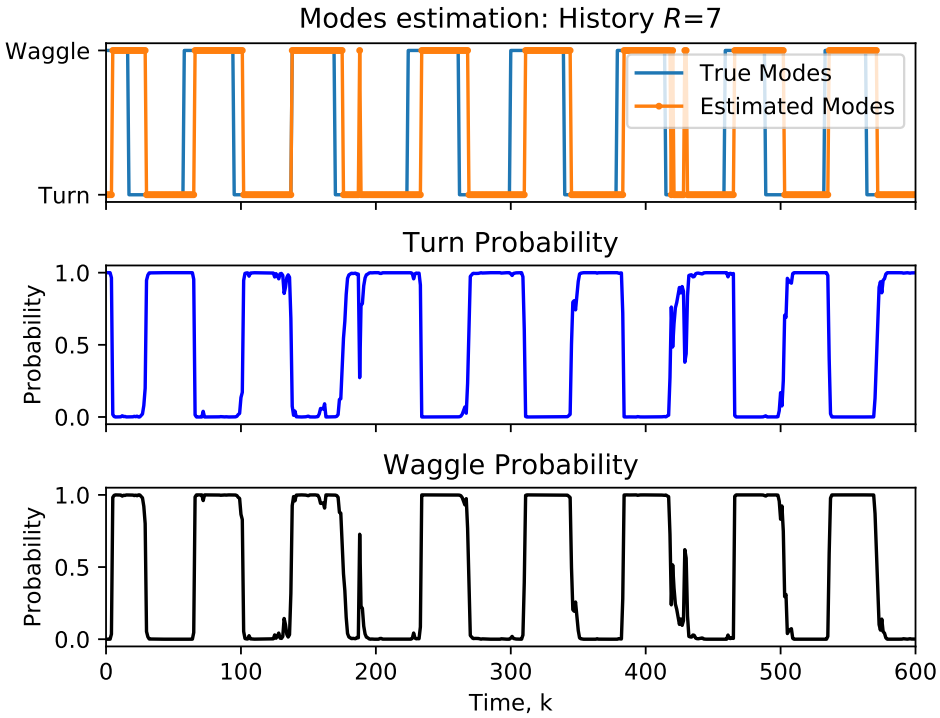


Figure 4.5: Estimation of Turn and Waggle modes for Sequence 6 of the Honeybee dataset. The upper plot shows the true and estimated modes. The middle plot shows the probability that the system will switch to turn mode at each instance of time, and the bottom plot shows the probability that the system will switch to waggle mode at each instance in time.

We make one observation from the above results. There is a lag of about nine samples between true mode and estimated mode. This condition is expected behavior for this dataset as there is no abrupt transition from waggle to turn and vice versa and we are using seven history points, our system will take some time to detect statistics of new mode from the measurement data.

The accuracy and error metrics are represented by a confusion matrix diagram and it is given by the following figure

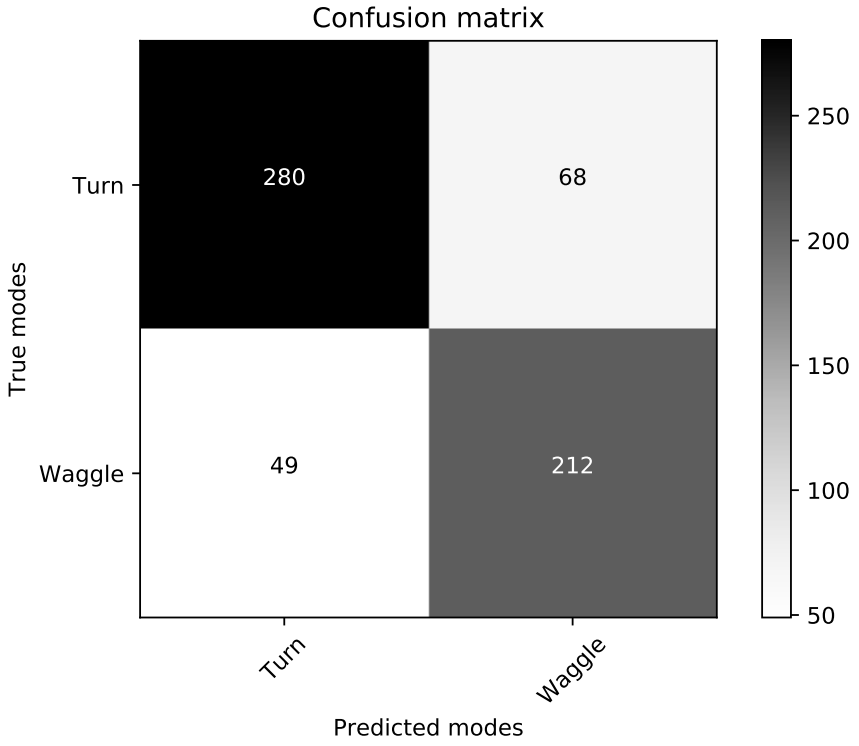


Figure 4.6: A comparison of true modes and predicted modes using confusion matrix diagram. The diagonal elements in the figure represent correct predictions and off-diagonal elements are the error of the system.

Using $R = 7$, our algorithm gives about 81% correct mode predictions and due to the lag in the system to promptly detect the mode change our system miss correct mode prediction about 19% of the time.

To evaluate state estimation performance, we compare the SNNT algorithm with the optimal Kalman filter obtained by learning the optimal state transimtion matrix as described in Section 4.1.2. The Euclidean distance between the true states and the estimated states gives the prediction error as a function of time index k. These error metrics are computed for the KF and SNNT algorithm and are presented in Figure 4.7 below.

We make two observations from the above figure. First, our algorithm is highly superior at tracking states of a dynamical system when compared to the Kalman filter algorithm. Using learned dynamics, SNNT has 91% lower RMSE than the optimally trained KF. Second, the fluctuations in error is more pronounced in KF than our algorithm. The improve-

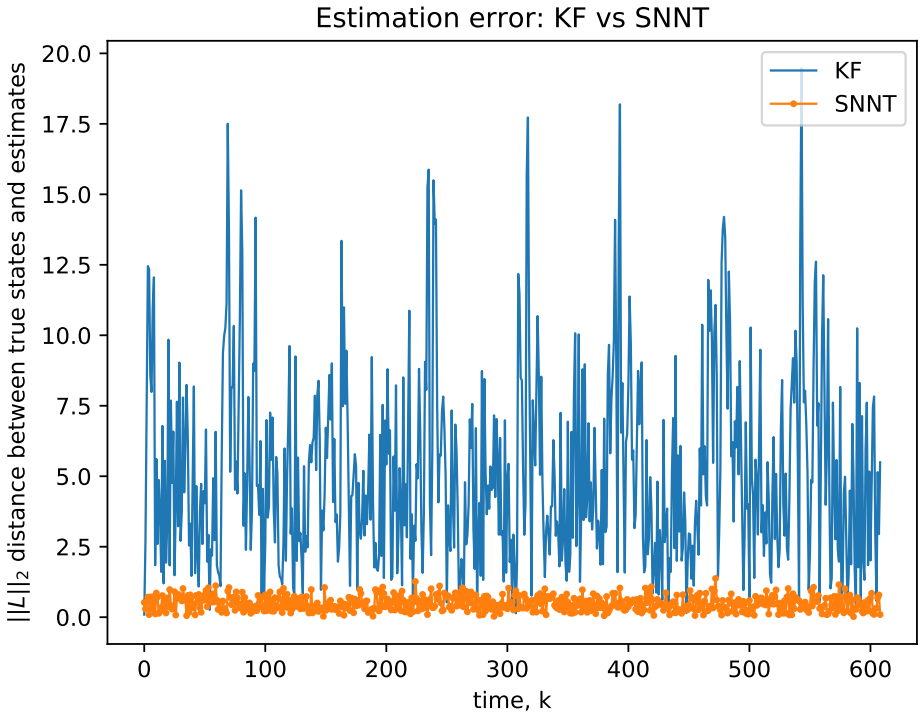


Figure 4.7: A comparison of estimation error for $Sequence$ 6 of the honeybee dataset using KF and SNNT.

ment in performance of tracking is due to the difference in working mechanism of two algorithms. Since our algorithm uses switching neural networks to model the transitions, and it can easily model nonlinear dynamics.

4.2 Simulated Robot Motion Dataset

4.2.1 Dataset description

The simulated dataset is generated considering two motion pattern of a line following robot, they are straight and turn with mode assignment $z_k = 0$ for turning motion and $z_k = 1$ for straight motion. For simulation purpose first, labels or modes are created using Markovian

dynamics described in Section 2.3 with transition probabilities:

$$\boldsymbol{T} = \begin{bmatrix} 0.90 & 0.10 \\ 0.12 & 0.88 \end{bmatrix}, \tag{4.2}$$

where the first row is for turning motion of the robot, and the second row is for the straight motion of the robot.

After generating N modes $z_1, \ldots, z_N$ for the system, we need to generate the state sequence $x_1, \ldots, x_N$. For this, we need two dynamic models and thus design two functions F_{turn} and $F_{straight}$ for turn motion and straight motion, respectively. F_{turn} transforms the given state to next state in a circular motion with a radius of 4 units and turn angle of $35°$. And $F_{straight}$ transforms the given state to next state in a linear path with distance between each transitions being 4 units. We use these two functions to simulate and generate points of N dimension using modes and transitions obtained for the system. We used Gaussian process noise with standard deviation $\sigma_p = 0.09$ and Gaussian measurement noise with standard deviation $\sigma_n = 0.8$. An example trajectory is shown in Figure 1.1.

4.2.2 Training details

Since there are two modes, we need two neural network functions, for this, we separate coordinates falling in two different modes keeping the relative position of the coordinates in the path. After this, for a history $R = 3$, the procedure discussed in Section 3.1.2 is followed to get the features for neural network training. The features set and target set thus generated are concatenated to get training sets $\mathbf{X}_{turn}$, $\mathbf{y}_{turn}$ associated with turning motion and $\mathbf{X}_{straight}$, $\mathbf{y}_{straight}$ associated with going straight. Mode $z_k = 0$ is assigned for the turn and mode $z_k = 1$ is assigned for straight. After training two separate neural networks as described in Section 3.1.1 we obtain two functions $\hat{F}_{turn}$ and $\hat{F}_{straight}$ for the turn and straight models respectively.

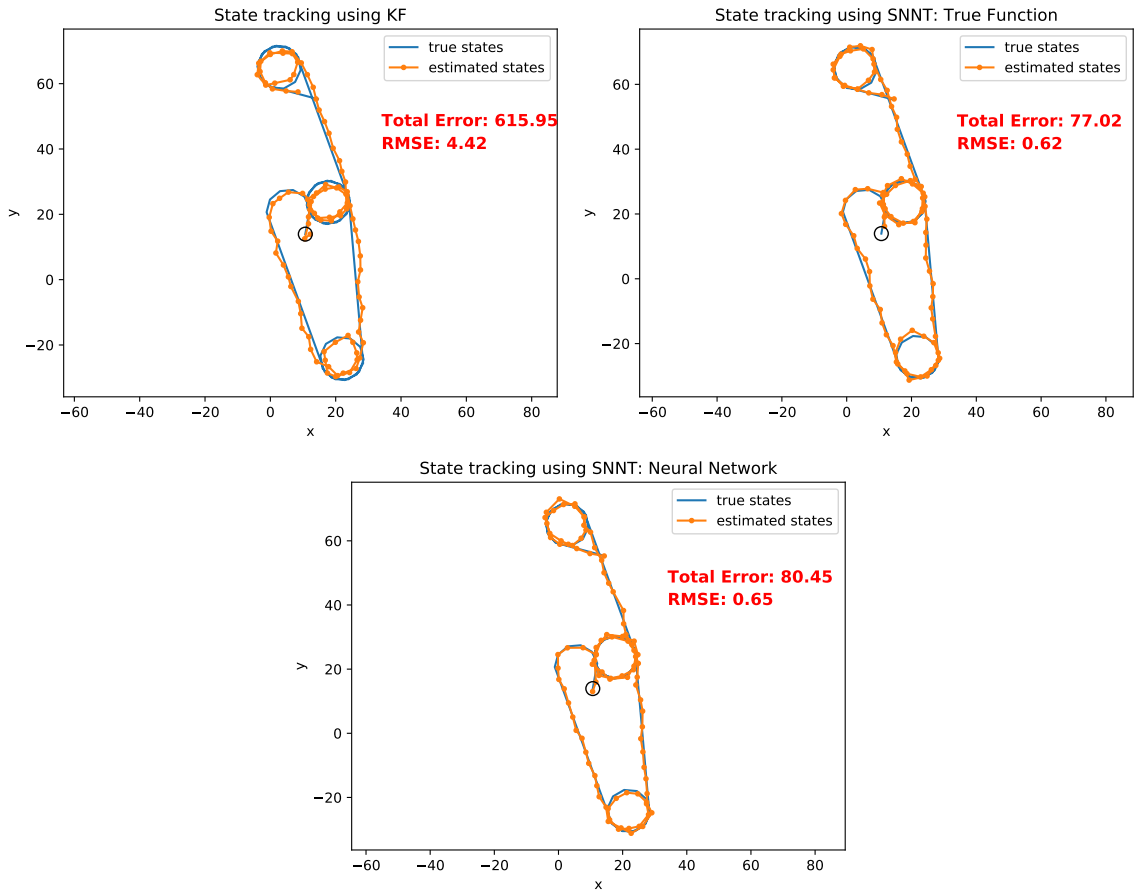


Figure 4.8: Robot tracking performance for 1) optimal Kalman filter, 2) SNNT using true dynamics F_{turn} and $F_{straight}$, 3) SNNT using neural network learned dynamics $\hat{F}_{turn}$ and $\hat{F}_{straight}$. The Switching Neural Network Tracker has lower state estimation error in both cases.

4.2.3 Results

In Figure 4.8, we compare state estimation error between the optimal Kalman filter (described in Section 4.1.2) and the SNNT algorithm using neural networks for learned motion dynamics. As an additional point of comparison, we also consider the performance of SNNT using the true generative (not learned) F_{turn} and $F_{straight}$ functions. This figure shows the superior performance of the SNNT algorithm when compared with the KF. Using learned dynamics, SNNT has 85% lower RMSE than the optimally trained KF.

In Figure 4.9 we plot the Euclidean error versus time for the same three estimators.

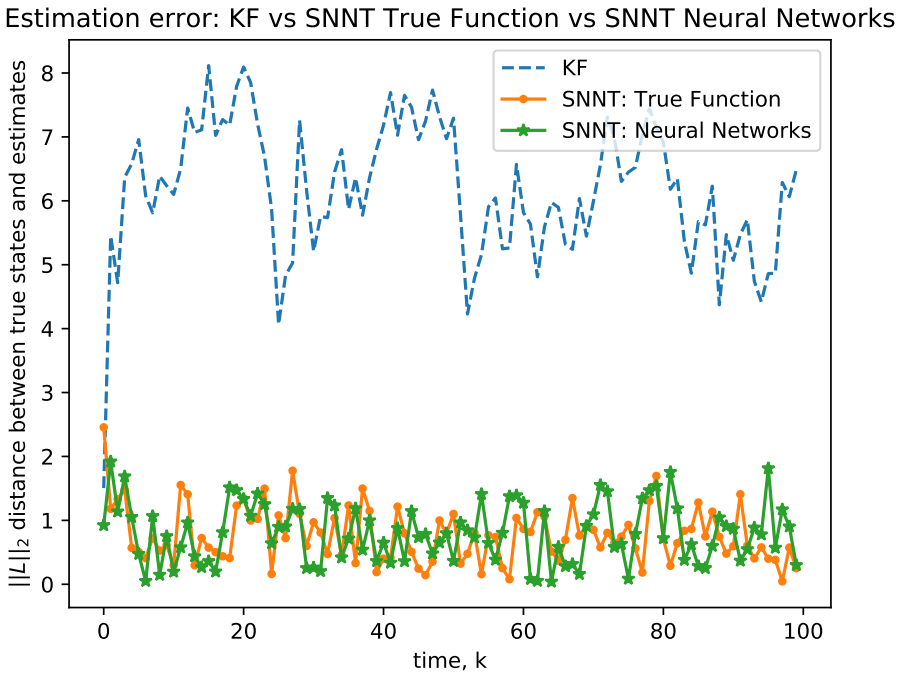


Figure 4.9: Estimation error comparison between Kalman filter and SNNT algorithms. In SNNT algorithm we use true function and neural network to model the transitions of states.

The figure indicates that Kalman filter state estimation error is, on average, approximately 6 units. In contrast, the SNNT algorithms maintain a per-instance error of approximately 1 unit. The neural network and true-fuction variants have approximately the same error, indicating that the neural network has appropriately learned the motion dynamics.

Finally, we consider mode estimation using SNNT. We again consider learned neural network and true-function variants, using $R = 3$ history points for both. True and estimated modes for the true functions (F_{turn} and $F_{straight}$) are given in Figure 4.10. The top plot of this figure shows the true mode and estimated mode at each time step k. The middle plot shows the probability that simulated robotic system will be in turn mode, and the bottom plot shows the probability that the robotic system will be in straight mode. Accuracy is measured using the count of same modes overlapping between true modes and estimated modes across all time instances. The correct mode estimation for SNNT, using true generative dynamics, was found to be 95%.

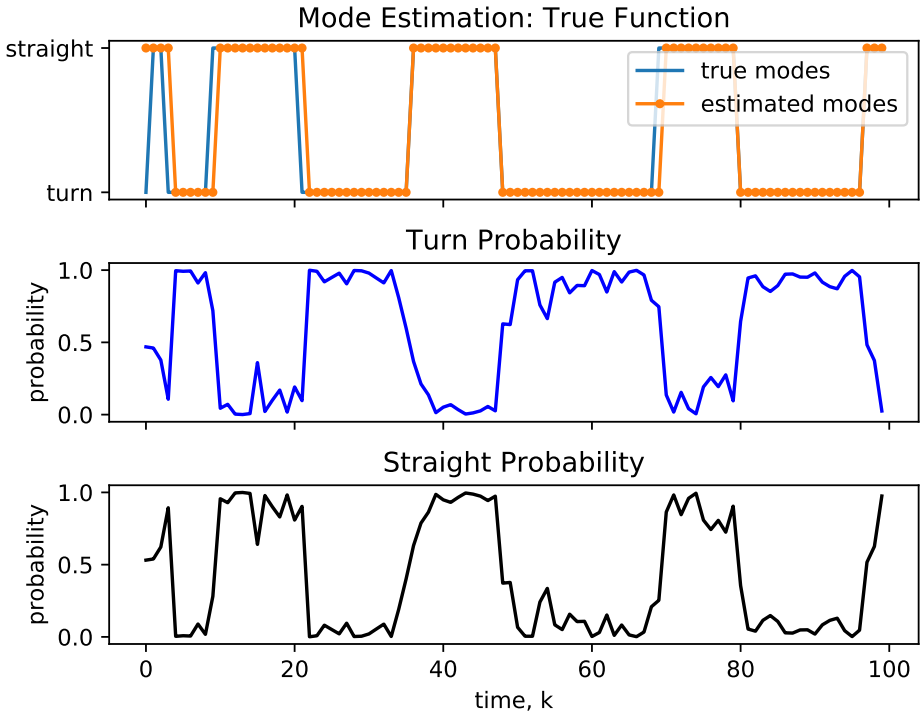


Figure 4.10: Mode estimation using SNNT and known generative dynamics F_{turn} and $F_{straight}$. The upper plot shows the true and estimated modes. The middle plot shows the probability that the system will switch to turn mode at time k, and the bottom figure shows the probability of switching to the straight mode at time k.

Mode estimates using neural networks for the learned dynamics $\hat{F}_{turn}$ and $\hat{F}_{straight}$ are shown in Figure 4.11. The SNNT algorithm, using learned dynamics, exhibited correct mode prediction 89% of the time. As expected, when the dynamics had to be learned, the performance was slightly worse.

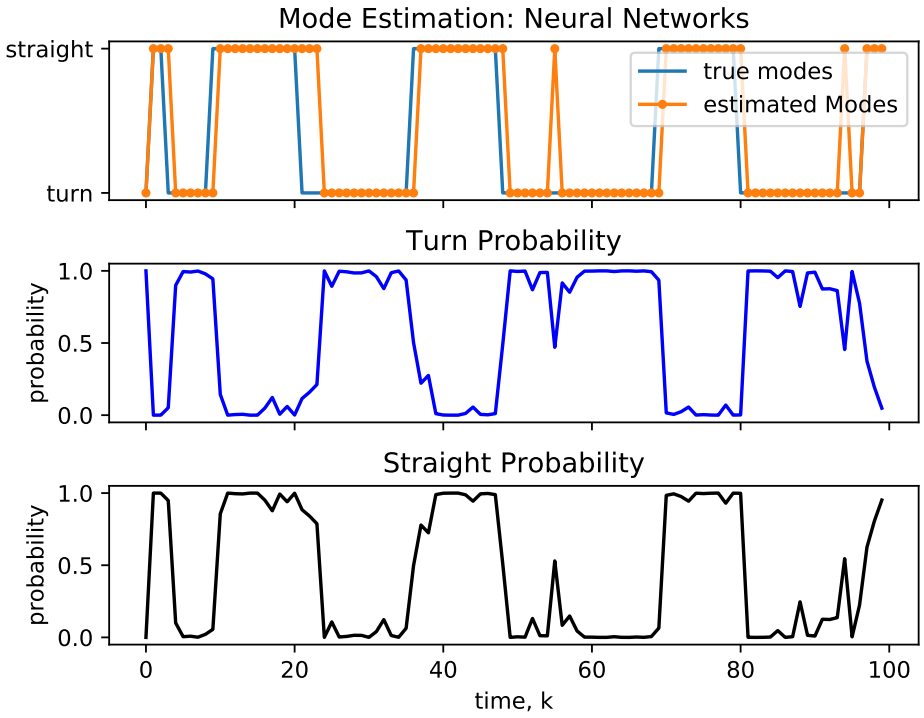


Figure 4.11: Mode estimation using SNNT and learned dynamics $\hat{F}_{turn}$ and $\hat{F}_{straight}$. The upper plot shows the true and estimated modes. The middle plot shows the probability that the system will switch to turn mode at time k, and the bottom figure shows the probability of switching to the straight mode at time k.

Conclusions

Many real-world systems have complex dynamic behavior that cannot be accurately modeled by linear or even switching linear systems. In this thesis, we developed an innovative model and tracking algorithm based on switching non-linear dynamics. Our model employed Markovian dynamics to model mode transitions within the system and used neural networks to capture non-linear dynamics within each mode of operation. We used Bayesian estimation methods to produce posterior mode and state estimates while employing the Unscented Transform to combat computational complexity.

We evaluated our algorithm on both measured and synthetic data. The measured dataset included the modal behavior of honeybees during their waggle dance. The proposed algorithm, dubbed the Switching Neural Network Tracker (SNNT), was successful in identifying the proper modes of motion and reduced state tracking error by 91%, relative to an optimally trained Kalman filter. Simulated data considered a maneuvering robot where, again, the SNNT demonstrated significantly better state estimates for this nonlinear dynamical system. Proper modes were identified 89% of the time and tracking error was reduced 85% relative to the Kalman filter.

Future work will compare the SNNT algorithm with recent tracking strategies from machine learning, such as long short-term memory (LSTM) networks. We will also consider strategies for simultaneous on-line model learning and tracking within SNNT.

Bibliography

[1] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, Feb 2002.

[2] H. Baltzakis and P. Trahanias. Hybrid mobile robot localization using switching state-space models. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, volume 1, pages 366–373 vol.1, May 2002.

[3] B. S. Chen, C. Y. Yang, F. K. Liao, and J. F. Liao. Mobile location estimator in a rough wireless environment using extended kalman-based imm and data fusion. *IEEE Transactions on Vehicular Technology*, 58(3):1157–1169, March 2009.

[4] S. Y. Chen. Kalman filter for robot vision: A survey. *IEEE Transactions on Industrial Electronics*, 59(11):4409–4420, Nov 2012.

[5] François Chollet et al. Keras. https://keras.io, 2015.

[6] Anna Dornhaus and Lars Chittka. Why do honey bees dance? *Behavioral Ecology and Sociobiology*, 55(4):395–401, 2004.

[7] E. Fox, E. B. Sudderth, M. I. Jordan, and A. S. Willsky. Bayesian nonparametric inference of switching dynamic linear models. *IEEE Transactions on Signal Processing*, 59(4):1569–1585, April 2011.

[8] S. Gannot, D. Burshtein, and E. Weinstein. Iterative and sequential kalman filter-based speech enhancement algorithms. *IEEE Transactions on Speech and Audio Processing*, 6(4):373–385, Jul 1998.

[9] Zoubin Ghahramani and Geoffrey E. Hinton. Variational learning for switching state-space models. *Neural Computation*, 12(4):831–864, 2000.

[10] M. S. Grewal and A. P. Andrews. Applications of kalman filtering in aerospace 1960 to the present [historical perspectives]. *IEEE Control Systems*, 30(3):69–78, June 2010.

[11] A. Gupta and B. Dhingra. Stock market prediction using hidden markov models. In *2012 Students Conference on Engineering and Systems*, pages 1–4, March 2012.

[12] L. Jin-Yue and Z. Bao-Ling. Research on the non-linear function fitting of rbf neural network. In *2013 International Conference on Computational and Information Sciences*, pages 842–845, June 2013.

[13] Simon J. Julier and Jeffrey K. Uhlmann. A new extension of the kalman filter to nonlinear systems. pages 182–193, 1997.

[14] R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, Mar 1960.

[15] S. Liang and R. Srikant. Why Deep Neural Networks for Function Approximation? *ArXiv e-prints*, October 2016.

[16] S. W. Linderman, A. C. Miller, R. P. Adams, D. M. Blei, L. Paninski, and M. J. Johnson. Recurrent switching linear dynamical systems. *ArXiv e-prints*, October 2016.

[17] V. Manfredi, S. Mahadevan, and J. Kurose. Switching kalman filters for prediction and tracking in an adaptive meteorological sensing network. In *2005 Second Annual*

IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, 2005. IEEE SECON 2005., pages 197–206, Sept 2005.

[18] Peter S. Maybeck. *Stochastic models, estimation, and control*, volume 141 of *Mathematics in Science and Engineering*. 1979.

[19] B. Mesot and D. Barber. Switching linear dynamical systems for noise robust speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(6):1850–1858, Aug 2007.

[20] Kevin P. Murphy. Switching kalman filters. Technical report, 1998.

[21] B. B. Nair, V. P. Mohandas, N. R. Sakthivel, S. Nagendran, A. Nareash, R. Nishanth, S. Ramkumar, and D. Manoj kumar. Application of hybrid adaptive filters for stock market prediction. In *2010 International Conference on Communication and Computational Intelligence (INCOCCI)*, pages 443–447, Dec 2010.

[22] Nguyet Nguyen and Dung Nguyen. Hidden markov model for stock selection. *Risks*, 3(4):455–473, 2015.

[23] Sang Min Oh, James M. Rehg, Tucker Balch, and Frank Dellaert. Learning and inferring motion patterns using parametric segmental switching linear dynamic systems. *International Journal of Computer Vision*, 77(1):103–124, May 2008.

[24] Ruey S. Tsay. *Analysis of Financial Time Series*. Wiley series in probability and statistics. 2010.

[25] Rudolph van der Merwe and Eric A. Wan. Sigma-point kalman filters for nonlinear estimation and sensor-fusion-applications to integrated navigation. 2000.

[26] E. A. Wan and R. Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*, pages 153–158, 2000.

[27] L. Wang. Dynamical models of stock prices based on technical trading rules part i: The models. *IEEE Transactions on Fuzzy Systems*, 23(4):787–801, Aug 2015.